ACL-05

43rd Annual Meeting of the Association for Computational Linguistics

Proceedings of the Conference

25-30 June 2005
University of Michigan

Production and Manufacturing by
Omnipress Inc.
Post Office Box 7214
Madison, WI 53707-7214

Preface: General Chair

When I asked what General Chairs are supposed to worry about, the main advice I got was this: Make sure the Program Co-Chairs are in the same time zone. Well, what's fun about that? Between Istanbul, Singapore, and Los Angeles, we could easily solve problems in real time; by the time Kemal arrived at the office after breakfast, Hwee Tou was just back from lunch, and I was done carrying out the late-night raid on my own refrigerator. No problem.

I'd like to start by thanking everyone who submitted research work to ACL-05. I'd especially like to thank researchers new to the field – this is a great time to be in computational linguistics. Excellent research is one of the Two Critical Ingredients of a successful ACL conference.

Program Co-Chairs **Hwee Tou Ng** and **Kemal Oflazer** deserve our gratitude for putting an immense amount of work into the main session program. They and the Area Chairs got a large number of submissions this year, and the program is diverse and exciting. Thanks also to **Erika Barragan-Nunez** for arranging the program committee meeting in California.

Stefan Riezler assembled a program of five excellent tutorials to begin the meeting, and **Mirella Lapata** organized the workshop program, assisted by **Mark Dras**, **Mary Harper**, **Dan Klein**, and **Shuly Winter**. **Masaaki Nagata** and **Ted Pedersen** put together a high-quality demo session, including software systems from all over the world.

Jason Eisner and **Philipp Koehn** put in a tremendous amount of thought, effort, and persistence into publications. Each time ACL doubles the number of papers, the work way more than doubles. **Mark Johnson**, as sponsorship chair, requested that the money be shown to him (and it was!), so thanks very much to the sponsors, and to Mark. **Richard Wicentowski** took on two chair roles – exhibits and publicity – the latter of which included writing the useful ACL-05 newsletters forwarded by ubiquifamous **Priscilla Rasmussen**.

Regina Barzilay, **Chris Callison-Burch** and **Stephen Wan** organized the Student Research Workshop (and thanks again to all the students who submitted their research). **Richard Power** graciously agreed to do pre-submission mentoring for authors. The ACL Executive Committee provided help on a number of issues and responded quickly to questions – thank you, **Martha Palmer**, **Jun'ichi Tsujii**, **Mark Steedman**, **Kathy McCoy**, **Sandee Carberry**, **Johanna Moore**, **Priscilla Rasmussen**, **Annie Zaenen**, **Walter Daelemans**, and **Keh-Yih Su**.

Dragomir Radev went far beyond the call of duty as Local Arrangements Chair. He raised and solved lots of strategic issues, followed up on every wire and cable, and cajoled other ACL chairs into solving important problems fast. I believe he may even be responsible for the weather and for making sure your luggage arrived on the same day you did. Thanks to the whole local team: **Rich Thomason**, **Steve Abney**, **Joyce Chai**, **San Duanmu**, **Kurt Godden**, **Acrisio Pires**, **Martha Pollack**, **Keith van der Linden**, **Rick Lewis**, **Sara Schwartz**, and **Bill Vlisides**, and to **James Sweeney**, who served as the conference webmaster. On behalf of Dragomir, please let me thank the University of Michigan's School of Information, Department of Electrical Engineering and Computer Science, and Department of Linguistics for their support. Dragomir also arranged the banquet at the Henry Ford Museum, where ACL President **Martha Palmer** will no doubt make an excellent speech – that's of course the Other Critical Ingredient of a successful ACL.

Finally, I'd also like to thank all the other folks who helped create ACL-05, including student volunteers, exhibitors, tutorialists, and everyone else not listed here.

To ACL attendees: thanks for coming, and please have a good conference!

Kevin Knight
ACL-05 General Chair
May 9, 2005

Preface: Program Co-Chairs

Exciting research in computational linguistics is being pursued vigorously all over the world. This year, we received a record number of 423 submissions. The program committee accepted 77 papers, for an acceptance rate of 18%, continuing the tradition of the annual ACL conference as being one of the most competitive and selective conferences. Of the accepted papers, 42 are from North America, 18 from Europe and the Middle East, and 17 from Asia and Australia.

We would like to express our heartfelt gratitude to all the authors who submitted their papers, to the 231 program committee members who worked tirelessly to review all submissions, and to the ten Area Chairs who oversaw the review process, collated the reviews, led discussions on papers with conflicting reviews, and solicited additional reviews for controversial papers. The Program Committee Co-Chairs and the area chairs then met for two days at the program committee meeting held at USC/ISI to select the final set of accepted papers. We would like to thank **Kevin Knight**, the General Conference Chair, who made available USC/ISI as the meeting venue, and his assistant **Erika Barragan-Nunez** who took care of the meeting arrangements and logistics.

The ACL-05 main program lasts three days, and includes plenary sessions, three parallel paper sessions, demo and poster sessions, and the student research workshop. We are grateful to Professor **Justine Cassell** (Northwestern University) and Professor **Michael Jordan** (University of California, Berkeley) who have kindly accepted our invitation to present invited talks at the conference.

The ACL-05 conference will also feature the ACL Lifetime Achievement Award. This prestigious award is presented to a most distinguished researcher for his or her pioneering work in computational linguistics. Past distinguished recipients of this award are **Aravind Joshi**, **Makoto Nagao**, and **Karen Spärck-Jones**. The recipient of this award in 2005 will be announced at a special plenary session at ACL-05, followed by a special lecture by the award recipient. ACL-05 will also continue the tradition of presenting the Best Paper Award to an outstanding paper. This award will be announced in the plenary session at the end of the conference.

A conference like ACL would not succeed without the many volunteers who offer their generous help. We deeply appreciate the advice and support of **Kevin Knight**, General Conference Chair, **Dragomir Radev**, Local Arrangements Chair, and the Local Arrangements Committee. We are also grateful to the ACL Executive Committee for their guidance, and **Walter Daelemans** and **Marilyn Walker**, ACL-04 Program Co-Chairs, for sharing their experience. We would also like to thank **Jason Eisner** and **Philipp Koehn**, Publication Co-Chairs, for putting together the proceedings of this conference.

We wish you an enjoyable time at ACL-05!

Hwee Tou Ng and Kemal Oflazer
ACL-05 Program Co-Chairs
May 12, 2005

Organizers

General Conference Chair:
Kevin Knight, University of Southern California, USA

Program Chairs:
Hwee Tou Ng, National University of Singapore, Singapore
Kemal Oflazer, Sabancı University, Turkey

Tutorial Chair:
Stefan Riezler, PARC, USA

Workshop Chair:
Mirella Lapata, University of Edinburgh, UK

Workshop Committee:
Mark Dras, Macquarie University, Australia
Mary Harper, NSF and Purdue University, USA
Dan Klein, University of California, Berkeley, USA
Shuly Winter, University of Haifa, Israel

Demo Chairs:
Masaaki Nagata, ATR Labs, USA
Ted Pedersen, University of Minnesota, USA

Publication Chairs:
Jason Eisner, Johns Hopkins University, USA
Philipp Koehn, University of Edinburgh, UK

Sponsorshop Chair:
Mark Johnson, Brown University, USA

Exhibits and Publicity Chair:
Richard Wicentowski, Swarthmore College, USA

Student Research Workshop:
Regina Barzilay, Massachusetts Institute of Technology, USA
Chris Callison-Burch, University of Edinburgh, UK
Stephen Wan, Macquarie University, Australia

Local Organization Chair:
Dragomir Radev, University of Michigan, USA

Local Organization Committee:
Steve Abney, University of Michigan, USA
Joyce Chai, Michigan State University, USA

Program Committee

Chairs

Hwee Tou Ng, National University of Singapore, Singapore
Kemal Oflazer, Sabancı University, Turkey

Area Chairs

Michael Collins, Massachusetts Institute of Technology, USA
Marti Hearst, University of California, Berkeley, USA
Hang Li, Microsoft Research, China
Chin-Yew Lin, University of Southern California, USA
Yuji Matsumoto, Nara Institute of Technology, Japan
Diana McCarthy, University of Sussex, UK
Hermann Ney, RWTH Aachen, Germany
Gerald Penn, University of Toronto, Canada
Brian Roark, Oregon Health and Science University, USA
Michael Strube, EML Research, Germany

Program Committee Members

Steve Abney (University of Michigan, USA), Eneko Agirre (University of the Basque Country, Spain), Yasemin Altun (Toyota Technological Institute at Chicago, USA), Rie Ando (IBM Research, USA), Chinatsu Aone (SRA International, USA), Masayuki Asahara (NAIST, Japan)

Tim Baldwin (University of Melbourne, Australia), Srinivas Bangalore (AT&T Research, USA), Regina Barzilay (Massachusetts Institute of Technology, USA), Dan Bikel (IBM Research, USA), Jeff Bilmes (University of Washington, USA), Steven Bird (University of Melbourne, Australia), Johan Bos (University of Edinburgh, UK), Antal van den Bosch (Tilburg University, The Netherlands), Thorsten Brants (Google, USA), Chris Brew (Ohio State University, USA), Ted Briscoe (University of Cambridge, UK), Bill Byrne (University of Cambridge, UK), Donna Byron (Ohio State University, USA)

Mary Elaine Califf (Illinois State University, USA), Jean Carletta (University of Edinburgh, UK), Bob Carpenter (Alias-i, USA), John Carroll (University of Sussex, UK), Francisco Casacuberta (Polytechnic University of Valencia, Spain), Justine Cassell (Northwestern University, USA), Nick Cercone (Dalhousie University, Canada), Mauro Cettolo (ITC-IRST, Italy), Eugene Charniak (Brown University, USA), Ciprian Chelba (Microsoft Research, USA), Aitao Chen (Yahoo, USA), Hsin-Hsi Chen (National Taiwan University, Taiwan), Stanley Chen (IBM Research, USA), David Chiang (University of Maryland, USA), Jennifer Chu-Carroll (IBM Research, USA), Tat-Seng Chua (National University of Singapore, Singapore), Grace Chung (CNRI, USA), Ken Church (Microsoft Research, USA), Fabio Ciravegna (University of Sheffield, UK), Alexander Clark (University of London, UK), Stephen Clark (University of Oxford, UK), William Cohen (Carnegie Mellon University, USA), Mark Craven (University of Wisconsin, USA), Richard Crouch (PARC, USA), James Curran (University of Sydney, Australia)

Jan Daciuk (Gdansk University of Technology, Poland), Walter Daelemans (University of Antwerp,

Table of Contents

Conference Program

Sunday, June 26, 2005 (continued)

Sunday, June 26, 2005 (continued)

Session M2M: Semantics

1:30–2:00	*The Distributional Inclusion Hypotheses and Lexical Entailment* Maayan Geffet and Ido Dagan
2:00–2:30	*Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales* Bo Pang and Lillian Lee
2:30–3:00	*Inducing Ontological Co-occurrence Vectors* Patrick Pantel
3:00–3:30	*Extracting Semantic Orientations of Words using Spin Model* Hiroya Takamura, Takashi Inui and Manabu Okumura

Session M2B: Discourse

1:30–2:00	*Modeling Local Coherence: An Entity-Based Approach* Regina Barzilay and Mirella Lapata
2:00–2:30	*Modelling the Substitutability of Discourse Connectives* Ben Hutchinson
2:30–3:00	*Machine Learning for Coreference Resolution: From Local Classification to Global Ranking* Vincent Ng
3:00–3:30	*Improving Pronoun Resolution Using Statistics-Based Semantic Compatibility Information* Xiaofeng Yang, Jian Su and Chew Lim Tan
3:30–4:00	Break

A High-Performance Semi-Supervised Learning Method for Text Chunking

Rie Kubota Ando† **Tong Zhang**‡

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, U.S.A.

†rie1@us.ibm.com ‡tongz@us.ibm.com

Abstract

In machine learning, whether one can build a more accurate classifier by using unlabeled data (*semi-supervised learning*) is an important issue. Although a number of semi-supervised methods have been proposed, their effectiveness on NLP tasks is not always clear. This paper presents a novel semi-supervised method that employs a learning paradigm which we call *structural learning*. The idea is to find "what good classifiers are like" by learning from thousands of automatically generated auxiliary classification problems on unlabeled data. By doing so, the common predictive structure shared by the multiple classification problems can be discovered, which can then be used to improve performance on the target problem. The method produces performance higher than the previous best results on CoNLL'00 syntactic chunking and CoNLL'03 named entity chunking (English and German).

1 Introduction

In supervised learning applications, one can often find a large amount of unlabeled data without difficulty, while labeled data are costly to obtain. Therefore, a natural question is whether we can use unlabeled data to build a more accurate classifier, given the same amount of labeled data. This problem is often referred to as *semi-supervised learning*.

Although a number of semi-supervised methods have been proposed, their effectiveness on NLP tasks is not always clear. For example, *co-training* (Blum and Mitchell, 1998) automatically bootstraps labels, and such labels are not necessarily reliable (Pierce and Cardie, 2001). A related idea is to use *Expectation Maximization* (EM) to *impute* labels. Although useful under some circumstances, when a relatively large amount of labeled data is available, the procedure often degrades performance (e.g. Merialdo (1994)). A number of bootstrapping methods have been proposed for NLP tasks (e.g. Yarowsky (1995), Collins and Singer (1999), Riloff and Jones (1999)). But these typically assume a very small amount of labeled data and have not been shown to improve state-of-the-art performance when a large amount of labeled data is available.

Our goal has been to develop a general learning framework for reliably using unlabeled data to improve performance irrespective of the amount of labeled data available. It is exactly this important and difficult problem that we tackle here.

This paper presents a novel semi-supervised method that employs a learning framework called *structural learning* (Ando and Zhang, 2004), which seeks to discover shared *predictive structures* (i.e. what good classifiers for the task are like) through jointly learning multiple classification problems on unlabeled data. That is, we systematically create thousands of problems (called *auxiliary problems*) relevant to the target task using unlabeled data, and train classifiers from the automatically generated 'training data'. We learn the commonality (or structure) of such many classifiers relevant to the task, and use it to improve performance on the target task. One example of such auxiliary problems for *chunking* tasks is to 'mask' a word and predict whether it is "people" or not from the context, like language modeling. Another example is to predict the prediction of some classifier trained for the target task. These auxiliary classifiers can be adequately learned since we have very large amounts of 'training data' for them, which we automatically generate from a very large amount of unlabeled data.

The contributions of this paper are two-fold. First, we present a novel robust semi-supervised method based on a new learning model and its application to chunking tasks. Second, we report higher performance than the previous best results on syntactic chunking (the CoNLL'00 corpus) and named entity chunking (the CoNLL'03 English and German corpora). In particular, our results are obtained by using unlabeled data as the *only* additional resource while many of the top systems rely on hand-crafted resources such as large name gazetteers or even rule-based post-processing.

2 A Model for Learning Structures

This work uses a linear formulation of structural learning. We first briefly review a standard linear prediction model and then extend it for structural learning. We sketch an optimization algorithm using SVD and compare it to related methods.

2.1 Standard linear prediction model

In the standard formulation of supervised learning, we seek a *predictor* that maps an input vector $\mathbf{x} \in \mathcal{X}$ to the corresponding output $y \in \mathcal{Y}$. *Linear prediction models* are based on real-valued predictors of the form $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w}$ is called a *weight vector*. For binary problems, the sign of the linear prediction gives the class label. For k-way classification (with $k > 2$), a typical method is *winner takes all*, where we train one predictor per class and choose the class with the highest output value.

A frequently used method for finding an accurate predictor $\hat{f}$ is regularized *empirical risk minimization (ERM)*, which minimizes an empirical loss of the predictor (with regularization) on the n training examples $\{(\mathbf{X}_i, Y_i)\}$:

$$\hat{f} = \arg\min_f \left(\sum_{i=1}^{n} L(f(\mathbf{X}_i), Y_i) + r(f) \right) .$$

$L(\cdot)$ is a *loss function* to quantify the difference between the prediction $f(\mathbf{X}_i)$ and the true output Y_i, and $r(\cdot)$ is a regularization term to control the model complexity. ERM-based methods for discriminative learning are known to be effective for NLP tasks such as chunking (e.g. Kudoh and Matsumoto (2001), Zhang and Johnson (2003)).

2.2 Linear model for structural learning

We present a linear prediction model for structural learning, which extends the traditional model to multiple problems. Specifically, we assume that there exists a *low-dimensional predictive structure* shared by multiple prediction problems. We seek to discover this structure through *joint empirical risk minimization* over the multiple problems.

Consider m problems indexed by $\ell \in \{1, \ldots, m\}$, each with n_ℓ samples $(\mathbf{X}_i^\ell, Y_i^\ell)$ indexed by $i \in \{1, \ldots, n_\ell\}$. In our joint linear model, a predictor for problem ℓ takes the following form

$$f_\ell(\Theta, \mathbf{x}) = \mathbf{w}_\ell^T \mathbf{x} + \mathbf{v}_\ell^T \Theta \mathbf{x} \ , \quad \Theta\Theta^T = \mathbf{I} \ , \qquad (1)$$

where we use $\mathbf{I}$ to denote the identity matrix. Matrix Θ (whose rows are orthonormal) is the common *structure parameter* shared by all the problems; $\mathbf{w}_\ell$ and $\mathbf{v}_\ell$ are weight vectors specific to each prediction problem ℓ. The idea of this model is to discover a common low-dimensional predictive structure (shared by the m problems) parameterized by the projection matrix Θ. In this setting, the goal of structural learning may also be regarded as *learning a good feature map* $\Theta\mathbf{x}$ — a low-dimensional feature vector parameterized by Θ.

In joint ERM, we seek Θ (and weight vectors) that minimizes the empirical risk summed over all the problems:

$$[\hat{\Theta}, \{\hat{f}_\ell\}] = \arg\min_{\Theta, \{f_\ell\}} \sum_{\ell=1}^{m} \left(\sum_{i=1}^{n_\ell} \frac{L(f_\ell(\Theta, \mathbf{X}_i^\ell), Y_i^\ell)}{n_\ell} + r(f_\ell) \right) . \qquad (2)$$

It can be shown that using joint ERM, we can reliably estimate the optimal joint parameter Θ as long as m is large (even when each n_ℓ is small). This is the key reason why structural learning is effective. A formal PAC-style analysis can be found in (Ando and Zhang, 2004).

2.3 Alternating structure optimization (ASO)

The optimization problem (2) has a simple solution using SVD when we choose square regularization:

$r(f_\ell) = \lambda \|\mathbf{w}_\ell\|_2^2$, where the regularization parameter λ is given. For clarity, let $\mathbf{u}_\ell$ be a weight vector for problem ℓ such that: $\mathbf{u}_\ell = \mathbf{w}_\ell + \Theta^T \mathbf{v}_\ell$. Then, (2) becomes the minimization of the joint empirical risk written as:

$$\sum_{\ell=1}^{m} \left(\sum_{i=1}^{n_\ell} \frac{L(\mathbf{u}_\ell^T \mathbf{X}_i^\ell, Y_i^\ell)}{n_\ell} + \lambda \|\mathbf{u}_\ell - \Theta^T \mathbf{v}_\ell\|_2^2 \right) . \qquad (3)$$

This minimization can be approximately solved by the following alternating optimization procedure:

- Fix $(\Theta, \{\mathbf{v}_\ell\})$, and find m predictors $\{\mathbf{u}_\ell\}$ that minimizes the joint empirical risk (3).
- Fix m predictors $\{\mathbf{u}_\ell\}$, and find $(\Theta, \{\mathbf{v}_\ell\})$ that minimizes the joint empirical risk (3).
- Iterate until a convergence criterion is met.

In the first step, we train m predictors independently. It is the second step that couples all the problems. Its solution is given by the SVD (singular value decomposition) of the predictor matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$: the rows of the optimum Θ are given by the most significant *left singular vectors*[1] of $\mathbf{U}$. Intuitively, the optimum Θ captures the maximal commonality of the m predictors (each derived from $\mathbf{u}_\ell$). These m predictors are updated using the new structure matrix Θ in the next iteration, and the process repeats.

Figure 1 summarizes the algorithm sketched above, which we call the *alternating structure optimization (ASO)* algorithm. The formal derivation can be found in (Ando and Zhang, 2004).

```
Input: training data $\{(\mathbf{X}_i^\ell, Y_i^\ell)\}$ $(\ell = 1, \ldots, m)$
Parameters: dimension $h$ and regularization param $\lambda$
Output: matrix $\Theta$ with $h$ rows
Initialize: $\mathbf{u}_\ell = 0$ $(\ell = 1 \ldots m)$, and arbitrary $\Theta$
iterate
  for $\ell = 1$ to $m$ do
    With fixed $\Theta$ and $\mathbf{v}_\ell = \Theta \mathbf{u}_\ell$, solve for $\hat{\mathbf{w}}_\ell$:
      $\hat{\mathbf{w}}_\ell = \arg\min_{\mathbf{w}_\ell} \Big[ \sum_{i=1}^{n_\ell} \frac{L(\mathbf{w}_\ell^T \mathbf{X}_i^\ell + (\mathbf{v}_\ell^T \Theta) \mathbf{X}_i^\ell, Y_i^\ell)}{n_\ell}$
                 $+ \lambda \|\mathbf{w}_\ell\|_2^2 \Big]$
    Let $\mathbf{u}_\ell = \hat{\mathbf{w}}_\ell + \Theta^T \mathbf{v}_\ell$
  endfor
  Compute the SVD of $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$.
  Let the rows of $\Theta$ be the $h$ left singular vectors of $\mathbf{U}$
    corresponding to the $h$ largest singular values.
until converge
```

Figure 1: SVD-based Alternating Structure Optimization (SVD-ASO) Algorithm

2.4 Comparison with existing techniques

It is important to note that this SVD-based ASO (SVD-ASO) procedure is fundamentally different from the usual principle component analysis (PCA), which can be regarded as dimension reduction in the *data space* $\mathcal{X}$. By contrast, the dimension reduction performed in the SVD-ASO algorithm is on the *predictor space* (a set of predictors). This is possible because we observe multiple predictors from multiple learning tasks. If we regard the observed predictors as sample points of the predictor distribution in the predictor space (corrupted with estimation error, or noise), then SVD-ASO can be interpreted as finding the “principle components” (or commonality) of these predictors (i.e., “what good predictors are like”). Consequently the method *directly* looks for low-dimensional structures with the highest predictive power. By contrast, the principle components of input data in the data space (which PCA seeks) may *not* necessarily have the highest predictive power.

The above argument also applies to the feature generation from unlabeled data using LSI (e.g. Ando (2004)). Similarly, Miller et al. (2004) used word-cluster memberships induced from an unannotated corpus as features for named entity chunking. Our work is related but more general, because we can explore additional information from unlabeled data using many different auxiliary problems. Since Miller et al. (2004)’s experiments used a proprietary corpus, direct performance comparison is not possible. However, our preliminary implementation of the word clustering approach did not provide any improvement on our tasks. As we will see, our starting performance is already high. Therefore the additional information discovered by SVD-ASO appears crucial to achieve appreciable improvements.

3 Semi-supervised Learning Method

For semi-supervised learning, the idea is to *create* many auxiliary prediction problems (relevant to the task) from unlabeled data so that we can learn the

[1]In other words, Θ is computed so that the best low-rank approximation of $\mathbf{U}$ in the least square sense is obtained by projecting $\mathbf{U}$ onto the row space of Θ; see e.g. Golub and Loan (1996) for SVD.

shared structure Θ (useful for the task) using the ASO algorithm. In particular, we want to create auxiliary problems with the following properties:

- *Automatic labeling*: we need to automatically generate various "labeled" data for the auxiliary problems from unlabeled data.
- *Relevancy*: auxiliary problems should be related to the target problem. That is, they should share a certain predictive structure.

The final classifier for the target task is in the form of (1), a linear predictor for structural learning. We fix Θ (learned from unlabeled data through auxiliary problems) and optimize weight vectors $\mathbf{w}$ and $\mathbf{v}$ on the given labeled data. We summarize this semi-supervised learning procedure below.

1. Create training data $\widetilde{Z}_\ell = \{(\widetilde{\mathbf{X}}_j, \widetilde{Y}_j^\ell)\}$ for each auxiliary problem ℓ from unlabeled data $\{\widetilde{\mathbf{X}}_j\}$.
2. Compute Θ from $\{\widetilde{Z}_\ell\}$ through SVD-ASO.
3. Minimize the empirical risk on the labeled data: $\hat{f} = \arg\min_f \sum_{i=1}^n \frac{L(f(\Theta, \mathbf{X}_i), Y_i)}{n} + \lambda \|\mathbf{w}\|_2^2$, where $f(\Theta, \mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{v}^T\Theta\mathbf{x}$ as in (1).

3.1 Auxiliary problem creation

The idea is to discover useful features (which do not necessarily appear in the labeled data) from the unlabeled data through learning auxiliary problems. Clearly, auxiliary problems more closely related to the target problem will be more beneficial. However, even if some problems are less relevant, they will not degrade performance severely since they merely result in some irrelevant features (originated from irrelevant Θ-components), which ERM learners can cope with. On the other hand, potential gains from relevant auxiliary problems can be significant. In this sense, our method is robust.

We present two general strategies for generating useful auxiliary problems: one in a completely unsupervised fashion, and the other in a partially-supervised fashion.

3.1.1 Unsupervised strategy

In the first strategy, we regard some observable substructures of the input data $\mathcal{X}$ as auxiliary class labels, and try to predict these labels using other parts of the input data.

Ex 3.1 Predict words. *Create auxiliary problems by regarding the word at each position as an auxiliary label, which we want to predict from the context. For instance, predict whether a word is "Smith" or not from its context. This problem is relevant to, for instance, named entity chunking since knowing a word is "Smith" helps to predict whether it is part of a name. One binary classification problem can be created for each possible word value (e.g., "IBM", "he", "get", $\cdots$). Hence, many auxiliary problems can be obtained using this idea.*

More generally, given a feature representation of the input data, we may mask some features as unobserved, and learn classifiers to predict these 'masked' features based on other features that are not masked. The automatic-labeling requirement is satisfied since the auxiliary labels are observable to us. To create relevant problems, we should choose to (mask and) predict features that have good correlation to the target classes, such as words on text tagging/chunking tasks.

3.1.2 Partially-supervised strategy

The second strategy is motivated by co-training. We use two (or more) distinct feature maps: $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$. First, we train a classifier F_1 for the target task, using the feature map $\mathbf{\Phi}_1$ and the labeled data. The auxiliary tasks are to predict the behavior of this classifier F_1 (such as predicted labels) on the unlabeled data, by using the other feature map $\mathbf{\Phi}_2$. Note that unlike co-training, we only use the classifier as a means of creating auxiliary problems that meet the relevancy requirement, instead of using it to bootstrap labels.

Ex 3.2 Predict the top-k choices of the classifier. *Predict the combination of k (a few) classes to which F_1 assigns the highest output (confidence) values. For instance, predict whether F_1 assigns the highest confidence values to* CLASS1 *and* CLASS2 *in this order. By setting $k = 1$, the auxiliary task is simply to predict the label prediction of classifier F_1. By setting $k > 1$, fine-grained distinctions (related to intrinsic sub-classes of target classes) can be learned. From a c-way classification problem, $c!/(c-k)!$ binary prediction problems can be created.*

4 Algorithms Used in Experiments

Using auxiliary problems introduced above, we study the performance of our semi-supervised learning method on named entity chunking and syntactic chunking. This section describes the algorithmic aspects of the experimental framework. The task-specific setup is described in Sections 5 and 6.

4.1 Extension of the basic SVD-ASO algorithm

In our experiments, we use an extension of SVD-ASO. In NLP applications, features have natural grouping according to their types/origins such as 'current words', 'parts-of-speech on the right', and so forth. It is desirable to perform a localized optimization for each of such natural feature groups. Hence, we associate each feature group with a submatrix of structure matrix Θ. The optimization algorithm for this extension is essentially the same as SVD-ASO in Figure 1, but with the SVD step performed separately for each group. See (Ando and Zhang, 2004) for the precise formulation. In addition, we regularize only those components of $\mathbf{w}_\ell$ which correspond to the non-negative part of $\mathbf{u}_\ell$. The motivation is that positive weights are usually directly related to the target concept, while negative ones often yield much less specific information representing 'the others'. The resulting extension, in effect, only uses the positive components of $\mathbf{U}$ in the SVD computation.

4.2 Chunking algorithm, loss function, training algorithm, and parameter settings

As is commonly done, we encode chunk information into word tags to cast the chunking problem to that of sequential word tagging. We perform Viterbi-style decoding to choose the word tag sequence that maximizes the sum of tagging confidence values.

In all settings (including baseline methods), the loss function is a modification of the Huber's robust loss for regression: $L(p, y) = \max(0, 1 - py)^2$ if $py \geq -1$; and $-4py$ otherwise; with square regularization ($\lambda = 10^{-4}$). One may select other loss functions such as SVM or logistic regression. The specific choice is not important for the purpose of this paper. The training algorithm is *stochastic gradient descent*, which is argued to perform well for regularized convex ERM learning formulations (Zhang, 2004).

As we will show in Section 7.3, our formulation is relatively insensitive to the change in h (row-dimension of the structure matrix). We fix h (for each feature group) to 50, and use it in all settings.

The most time-consuming process is the training of m auxiliary predictors on the unlabeled data (computing $\mathbf{U}$ in Figure 1). Fixing the number of iterations to a constant, it runs in linear to m and the number of unlabeled instances and takes hours in our settings that use more than 20 million unlabeled instances.

4.3 Baseline algorithms

Supervised classifier For comparison, we train a classifier using the same features and algorithm, but without unlabeled data ($\Theta = \mathbf{0}$ in effect).

Co-training We test co-training since our idea of partially-supervised auxiliary problems is motivated by co-training. Our implementation follows the original work (Blum and Mitchell, 1998). The two (or more) classifiers (with distinct feature maps) are trained with labeled data. We maintain a pool of q unlabeled instances by random selection. The classifier proposes labels for the instances in this pool. We choose s instances for each classifier with high confidence while preserving the class distribution observed in the initial labeled data, and add them to the labeled data. The process is then repeated. We explore q=50K, 100K, s=50,100,500,1K, and commonly-used feature splits: 'current vs. context' and 'current+left-context vs. current+right-context'.

Self-training Single-view bootstrapping is sometimes called *self-training*. We test the basic self-training[2], which replaces multiple classifiers in the co-training procedure with a single classifier that employs all the features.

co/self-training oracle performance To avoid the issue of parameter selection for the co- and self-training, we report their best possible *oracle performance*, which is the best F-measure number among all the co- and self-training parameter settings including the choice of the number of iterations.

[2]We also tested "self-training with bagging", which Ng and Cardie (2003) used for co-reference resolution. We omit results since it did not produce better performance than the supervised baseline.

- words, parts-of-speech (POS), character types, 4 characters at the beginning/ending in a 5-word window.
- words in a 3-syntactic chunk window.
- labels assigned to two words on the left.
- bi-grams of the current word and the label on the left.
- labels assigned to previous occurrences of the current word.

Figure 2: Feature types for named entity chunking. POS and syntactic chunk information is provided by the organizer.

5 Named Entity Chunking Experiments

We report named entity chunking performance on the CoNLL'03 shared-task[3] corpora (English and German). We choose this task because the original intention of this shared task was to test the effectiveness of semi-supervised learning methods. However, it turned out that none of the top performing systems used unlabeled data. The likely reason is that the number of labeled data is relatively large (>200K), making it hard to benefit from unlabeled data. We show that our ASO-based semi-supervised learning method (hereafter, *ASO-semi*) can produce results appreciably better than all of the top systems, by using unlabeled data as the *only* additional resource. In particular, we do not use any gazetteer information, which was used in all other systems.

The CoNLL corpora are annotated with four types of named entities: persons, organizations, locations, and miscellaneous names (e.g., "World Cup"). We use the official training/development/test splits. Our unlabeled data sets consist of 27 million words (English) and 35 million words (German), respectively. They were chosen from the same sources – Reuters and ECI Multilingual Text Corpus – as the provided corpora but disjoint from them.

5.1 Features

Our feature representation is a slight modification of a simpler configuration (without any gazetteer) in (Zhang and Johnson, 2003), as shown in Figure 2. We use POS and syntactic chunk information provided by the organizer.

5.2 Auxiliary problems

As shown in Figure 3, we experiment with auxiliary problems from Ex 3.1 and 3.2: "Predict current (or previous or next) words"; and "Predict *top-2* choices of the classifier" using feature splits 'left context vs. the others' and 'right context vs. the others'. For word-prediction problems, we only consider the instances whose current words are either nouns or adjectives since named entities mostly consist of these types. Also, we leave out all but at most 1000 binary prediction problems of each type that have the largest numbers of positive examples to ensure that auxiliary predictors can be adequately learned with a sufficiently large number of examples. The results we report are obtained by using all the problems in Figure 3 unless otherwise specified.

[3]http://cnts.uia.ac.be/conll2003/ner

# of aux. problems	Auxiliary labels	Features used for learning aux problems
1000	previous words	all but previous words
1000	current words	all but current words
1000	next words	all but next words
72	F_1's top-2 choices	$\mathbf{\Phi}_2$ (all but left context)
72	F_2's top-2 choices	$\mathbf{\Phi}_1$ (left context)
72	F_3's top-2 choices	$\mathbf{\Phi}_4$ (all but right context)
72	F_4's top-2 choices	$\mathbf{\Phi}_3$ (right context)

Figure 3: Auxiliary problems used for named entity chunking. 3000 problems 'mask' words and predict them from the other features on unlabeled data. 288 problems predict classifier F_i's predictions on unlabeled data, where F_i is trained with labeled data using feature map $\mathbf{\Phi}_i$. There are 72 possible top-2 choices from 9 classes (beginning/inside of four types of name chunks and 'outside').

5.3 Named entity chunking results

methods	test data	F	diff. from supervised		
			prec.	recall	F
English, small (10K examples) training set					
ASO-semi	dev.	**81.25**	*+10.02*	*+7.00*	*+8.51*
co/self oracle		73.10	+0.32	+0.39	+0.36
ASO-semi	test	**78.42**	*+9.39*	*+10.73*	*+10.10*
co/self oracle		69.63	+0.60	+1.95	+1.31
English, all (204K) training examples					
ASO-semi	dev.	**93.15**	*+2.25*	*+3.00*	*+2.62*
co/self oracle		90.64	+0.04	+0.20	+0.11
ASO-semi	test	**89.31**	*+3.20*	*+4.51*	*+3.86*
co/self oracle		85.40	−0.04	−0.05	−0.05
German, all (207K) training examples					
ASO-semi	dev.	**74.06**	*+7.04*	*+10.19*	*+9.22*
co/self oracle		66.47	−2.59	+4.39	+1.63
ASO-semi	test	**75.27**	*+4.64*	*+6.59*	*+5.88*
co/self oracle		70.45	−1.26	+2.59	+1.06

Figure 4: Named entity chunking results. No gazetteer. F-measure and performance improvements over the supervised baseline in precision, recall, and F. For co- and self-training (baseline), the *oracle* performance is shown.

Figure 4 shows results in comparison with the supervised baseline in six configurations, each trained with one of three sets of labeled training examples: a small English set (10K examples randomly chosen), the entire English training set (204K), and the entire German set (207K), tested on either the development set or test set. ASO-semi significantly improves both precision and recall in all the six configurations, resulting in improved F-measures over the supervised baseline by +2.62% to +10.10%.

Co- and self-training, at their *oracle performance*, improve recall but often degrade precision; consequently, their F-measure improvements are relatively low: −0.05% to +1.63%.

Comparison with top systems As shown in Figure 5, ASO-semi achieves higher performance than the top systems on both English and German data. Most of the top systems boost performance by external hand-crafted resources such as: large gazetteers[4]; a large amount (2 million words) of *labeled* data manually annotated with finer-grained named entities (FIJZ03); and rule-based post processing (KSNM03). Hence, we feel that our results, obtained by using unlabeled data as the only additional resource, are encouraging.

System	Eng.	Ger.	Additional resources
ASO-semi	**89.31**	**75.27**	*unlabeled data*
FIJZ03	88.76	72.41	gazetteers; 2M-word labeled data (English)
CN03	88.31	65.67	gazetteers (English); (also very elaborated features)
KSNM03	86.31	71.90	rule-based post processing

Figure 5: Named entity chunking. F-measure on the test sets. Previous best results: FIJZ03 (Florian et al., 2003), CN03 (Chieu and Ng, 2003), KSNM03 (Klein et al., 2003).

6 Syntactic Chunking Experiments

Next, we report syntactic chunking performance on the CoNLL'00 shared-task[5] corpus. The training and test data sets consist of the Wall Street Journal corpus (WSJ) sections 15–18 (212K words) and section 20, respectively. They are annotated with eleven types of syntactic chunks such as noun phrases. We use the WSJ articles in 1991 (15 million words) from the TREC corpus as the unlabeled data.

[4]Whether or not gazetteers are useful depends on their coverage. A number of top-performing systems used their own gazetteers in addition to the organizer's gazetteers and reported significant performance improvements (e.g., FIJZ03, CN03, and ZJ03).

[5]http://cnts.uia.ac.be/conll2000/chunking

· uni- and bi-grams of words and POS in a 5-token window.
· word-POS bi-grams in a 3-token window.
· POS tri-grams on the left and right.
· labels of the two words on the left and their bi-grams.
· bi-grams of the current word and two labels on the left.

Figure 6: Feature types for syntactic chunking. POS information is provided by the organizer.

	prec.	recall	$F_{\beta=1}$
supervised	93.83	93.37	93.60
ASO-semi	**94.57**	**94.20**	**94.39 (+0.79)**
co/self oracle	93.76	93.56	93.66 (+0.06)

Figure 7: Syntactic chunking results.

6.1 Features and auxiliary problems

Our feature representation is a slight modification of a simpler configuration (without linguistic features) in (Zhang et al., 2002), as shown in Figure 6. We use the POS information provided by the organizer. The types of auxiliary problems are the same as in the named entity experiments. For word predictions, we exclude instances of punctuation symbols.

6.2 Syntactic chunking results

As shown in Figure 7, ASO-semi improves both precision and recall over the supervised baseline. It achieves 94.39% in F-measure, which outperforms the supervised baseline by 0.79%. Co- and self-training again slightly improve recall but slightly degrade precision at their oracle performance, which demonstrates that it is not easy to benefit from unlabeled data on this task.

Comparison with the previous best systems As shown in Figure 8, ASO-semi achieves performance higher than the previous best systems. Though the space constraint precludes providing the detail, we note that ASO-semi outperforms all of the previous top systems in both precision and recall. Unlike named entity chunking, the use of external resources on this task is rare. An exception is the use of output from a grammar-based full parser as features in ZDJ02+, which our system does not use. KM01 and CM03 boost performance by classifier combinations. SP03 trains conditional random fields for NP

	all	NP	description
ASO-semi	**94.39**	**94.70**	+*unlabeled data*
KM01	93.91	94.39	SVM combination
CM03	93.74	94.41	perceptron in two layers
SP03	–	94.38	conditional random fields
ZDJ02	93.57	93.89	generalized Winnow
ZDJ02+	94.17	94.38	+full parser output

Figure 8: Syntactic chunking F-measure. Comparison with previous best results: KM01 (Kudoh and Matsumoto, 2001), CM03 (Carreras and Marquez, 2003), SP03 (Sha and Pereira, 2003), ZDJ02 (Zhang et al., 2002).

(noun phrases) only. ASO-semi produces higher NP chunking performance than the others.

7 Empirical Analysis

7.1 Effectiveness of auxiliary problems

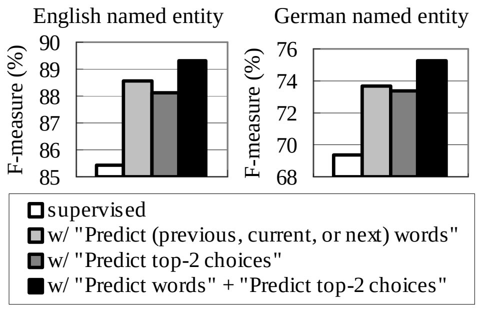


Figure 9: Named entity F-measure produced by using individual types of auxiliary problems. Trained with the entire training sets and tested on the test sets.

Figure 9 shows F-measure obtained by computing Θ from individual types of auxiliary problems on named entity chunking. Both types – “Predict words” and “Predict top-2 choices of the classifier” – are useful, producing significant performance improvements over the supervised baseline. The best performance is achieved when Θ is produced from all of the auxiliary problems.

7.2 Interpretation of Θ

To gain insights into the information obtained from unlabeled data, we examine the Θ entries associated with the feature ‘current words’, computed for the English named entity task. Figure 10 shows the features associated with the entries of Θ with the largest values, computed from the 2000 unsupervised auxiliary problems: “Predict previous words” and “Predict next words”. For clarity, the figure only shows

row#	Features corresponding to significant Θ entries	Interpretation
4	Ltd, Inc, Plc, International, Ltd., Association, Group, Inc.	organizations
7	Co, Corp, Co., Company, Authority, Corp., Services	organizations
9	PCT, N/A, Nil, Dec, BLN, Avg, Year-on-year, UNCH	no names
11	New, France, European, San, North, Japan, Asian, India	locations
15	Peter, Sir, Charles, Jose, Paul, Lee, Alan, Dan, John, James	persons
26	June, May, July, Jan, March, August, September, April	months

Figure 10: Interpretation of Θ computed from word-prediction (unsupervised) problems for named entity chunking.

words beginning with upper-case letters (i.e., likely to be names in English). Our method captures the spirit of predictive word-clustering but is more general and effective on our tasks.

It is possible to develop a general theory to show that the auxiliary problems we use are helpful under reasonable conditions. The intuition is as follows. Suppose we split the features into two parts $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ and predict $\mathbf{\Phi}_1$ based on $\mathbf{\Phi}_2$. Suppose features in $\mathbf{\Phi}_1$ are correlated to the class labels (but not necessarily correlated among themselves). Then, the auxiliary prediction problems are related to the target task, and thus can reveal useful structures of $\mathbf{\Phi}_2$. Under some conditions, it can be shown that features in $\mathbf{\Phi}_2$ with similar predictive performance tend to map to similar low-dimensional vectors through Θ. This effect can be empirically observed in Figure 10 and will be formally shown elsewhere.

7.3 Effect of the Θ dimension

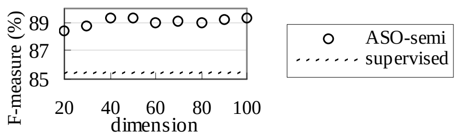


Figure 11: F-measure in relation to the row-dimension of Θ. English named entity chunking, test set.

Recall that throughout the experiments, we fix the row-dimension of Θ (for each feature group) to 50. Figure 11 plots F-measure in relation to the row-dimension of Θ, which shows that the method is relatively insensitive to the change of this parameter, at least in the range which we consider.

8 Conclusion

We presented a novel semi-supervised learning method that learns the most predictive low-dimensional feature projection from unlabeled data using the structural learning algorithm SVD-ASO. On CoNLL'00 syntactic chunking and CoNLL'03 named entity chunking (English and German), the method exceeds the previous best systems (including those which rely on hand-crafted resources) by using unlabeled data as the only additional resource.

The key idea is to create auxiliary problems automatically from unlabeled data so that predictive structures can be learned from that data. In practice, it is desirable to create as many auxiliary problems as possible, as long as there is some reason to believe in their relevancy to the task. This is because the risk is relatively minor while the potential gain from relevant problems is large. Moreover, the auxiliary problems used in our experiments are merely possible examples. One advantage of our approach is that one may design a variety of auxiliary problems to learn various aspects of the target problem from unlabeled data. Structural learning provides a framework for carrying out possible new ideas.

Acknowledgments

Part of the work was supported by ARDA under the NIMD program PNWD-SW-6059.

References

Rie Kubota Ando and Tong Zhang. 2004. A framework for learning predictive structures from multiple tasks and unlabeled data. Technical report, IBM. RC23462.

Rie Kubota Ando. 2004. Semantic lexicon construction: Learning from unlabeled data via spectral analysis. In *Proceedings of CoNLL-2004*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *proceedings of COLT-98*.

Xavier Carreras and Lluis Marquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of RANLP-2003*.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings CoNLL-2003*, pages 160–163.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of EMNLP/VLC'99*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings CoNLL-2003*, pages 168–171.

Gene H. Golub and Charles F. Van Loan. 1996. Matrix computations third edition.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. 2003. Named entity recognition with character-level models. In *Proceedings CoNLL-2003*, pages 188–191.

Taku Kudoh and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL 2001*.

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL-2004*.

Vincent Ng and Claire Cardie. 2003. Weakly supervised natural language learning without redundant views. In *Proceedings of HLT-NAACL-2003*.

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *Proceedings of EMNLP-2001*.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI-99*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL'03*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL-95*.

Tong Zhang and David E. Johnson. 2003. A robust risk minimization based named entity recognition system. In *Proceedings CoNLL-2003*, pages 204–207.

Tong Zhang, Fred Damerau, and David E. Johnson. 2002. Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research*, 2:615–637.

Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 04*, pages 919–926.

Scaling Conditional Random Fields Using Error-Correcting Codes

Trevor Cohn
Department of Computer Science
and Software Engineering
University of Melbourne, Australia
tacohn@csse.unimelb.edu.au

Andrew Smith
Division of Informatics
University of Edinburgh
United Kingdom
a.p.smith-2@sms.ed.ac.uk

Miles Osborne
Division of Informatics
University of Edinburgh
United Kingdom
miles@inf.ed.ac.uk

Abstract

Conditional Random Fields (CRFs) have been applied with considerable success to a number of natural language processing tasks. However, these tasks have mostly involved very small label sets. When deployed on tasks with larger label sets, the requirements for computational resources mean that training becomes intractable.

This paper describes a method for training CRFs on such tasks, using error correcting output codes (ECOC). A number of CRFs are independently trained on the separate binary labelling tasks of distinguishing between a subset of the labels and its complement. During decoding, these models are combined to produce a predicted label sequence which is resilient to errors by individual models.

Error-correcting CRF training is much less resource intensive and has a much faster training time than a standardly formulated CRF, while decoding performance remains quite comparable. This allows us to scale CRFs to previously impossible tasks, as demonstrated by our experiments with large label sets.

1 Introduction

Conditional random fields (CRFs) (Lafferty et al., 2001) are probabilistic models for labelling sequential data. CRFs are undirected graphical models that define a conditional distribution over label sequences given an observation sequence. They allow the use of arbitrary, overlapping, non-independent features as a result of their global conditioning. This allows us to avoid making unwarranted independence assumptions over the observation sequence, such as those required by typical generative models.

Efficient inference and training methods exist when the graphical structure of the model forms a chain, where each position in a sequence is connected to its adjacent positions. CRFs have been applied with impressive empirical results to the tasks of named entity recognition (McCallum and Li, 2003), simplified part-of-speech (POS) tagging (Lafferty et al., 2001), noun phrase chunking (Sha and Pereira, 2003) and extraction of tabular data (Pinto et al., 2003), among other tasks.

CRFs are usually estimated using gradient-based methods such as limited memory variable metric (LMVM). However, even with these efficient methods, training can be slow. Consequently, most of the tasks to which CRFs have been applied are relatively small scale, having only a small number of training examples and small label sets. For much larger tasks, with hundreds of labels and millions of examples, current training methods prove intractable. Although training can potentially be parallelised and thus run more quickly on large clusters of computers, this in itself is not a solution to the problem: tasks can reasonably be expected to increase in size and complexity much faster than any increase in computing power. In order to provide scalability, the factors which most affect the resource usage and runtime of the training method must be addressed directly – ideally the dependence on the number of labels should be reduced.

This paper presents an approach which enables CRFs to be used on larger tasks, with a significant reduction in the time and resources needed for training. This reduction does not come at the cost of performance – the results obtained on benchmark natural language problems compare favourably, and sometimes exceed, the results produced from regular CRF training. *Error correcting output codes* (ECOC) (Dietterich and Bakiri, 1995) are used to train a community of CRFs on binary tasks, with each discriminating between a subset of the labels and its complement. Inference is performed by applying these 'weak' models to an unknown example, with each component model removing some ambiguity when predicting the label sequence. Given a sufficient number of binary models predicting suitably diverse label subsets, the label sequence can be inferred while being robust to a number of individual errors from the weak models. As each of these weak models are binary, individually they can be efficiently trained, even on large problems. The number of weak learners required to achieve good performance is shown to be relatively small on practical tasks, such that the overall complexity of error-correcting CRF training is found to be much less than that of regular CRF training methods.

We have evaluated the error-correcting CRF on the CoNLL 2003 named entity recognition (NER) task (Sang and Meulder, 2003), where we show that the method yields similar generalisation performance to standardly formulated CRFs, while requiring only a fraction of the resources, and no increase in training time. We have also shown how the error-correcting CRF scales when applied to the larger task of POS tagging the Penn Treebank and also the even larger task of simultaneously noun phrase chunking (NPC) and POS tagging using the CoNLL 2000 data-set (Sang and Buchholz, 2000).

2 Conditional random fields

CRFs are undirected graphical models used to specify the conditional probability of an assignment of output labels given a set of input observations. We consider only the case where the output labels of the model are connected by edges to form a linear chain. The joint distribution of the label sequence, $\mathbf{y}$, given the input observation sequence, $\mathbf{x}$, is given by

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \sum_{t=1}^{T+1} \sum_{k} \lambda_k f_k(t, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{x})$$

where T is the length of both sequences and λ_k are the parameters of the model. The functions f_k are feature functions which map properties of the observation and the labelling into a scalar value. $Z(\mathbf{x})$ is the partition function which ensures that p is a probability distribution.

A number of algorithms can be used to find the optimal parameter values by maximising the log-likelihood of the training data. Assuming that the training sequences are drawn *IID* from the population, the conditional log likelihood $\mathcal{L}$ is given by

$$\begin{aligned}\mathcal{L} &= \sum_i \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \\ &= \sum_i \left\{ \sum_{t=1}^{T^{(i)}+1} \sum_k \lambda_k f_k(t, \mathbf{y}_{t-1}^{(i)}, \mathbf{y}_t^{(i)}, \mathbf{x}^{(i)}) \right. \\ &- \left. \log Z(\mathbf{x}^{(i)}) \right\}\end{aligned}$$

where $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ are the i^{th} observation and label sequence. Note that a prior is often included in the $\mathcal{L}$ formulation; it has been excluded here for clarity of exposition. CRF estimation methods include generalised iterative scaling (GIS), improved iterative scaling (IIS) and a variety of gradient based methods. In recent empirical studies on maximum entropy models and CRFs, limited memory variable metric (LMVM) has proven to be the most efficient method (Malouf, 2002; Wallach, 2002); accordingly, we have used LMVM for CRF estimation.

Every iteration of LMVM training requires the computation of the log-likelihood and its derivative with respect to each parameter. The partition function $Z(\mathbf{x})$ can be calculated efficiently using dynamic programming with the forward algorithm. $Z(\mathbf{x})$ is given by $\sum_y \alpha_T(y)$ where α are the forward values, defined recursively as

$$\alpha_{t+1}(y) = \sum_{y'} \alpha_t(y') \exp \sum_k \lambda_k f_k(t+1, y', y, \mathbf{x})$$

The derivative of the log-likelihood is given by

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \sum_i \left\{ \sum_{t=1}^{T^{(i)}+1} f_k(t, \mathbf{y}_{t-1}^{(i)}, \mathbf{y}_t^{(i)}, \mathbf{x}^{(i)}) - \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}^{(i)}) \sum_{t=1}^{T^{(i)}+1} f_k(t, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{x}^{(i)}) \right\}$$

The first term is the empirical count of feature k, and the second is the expected count of the feature under the model. When the derivative equals zero – at convergence – these two terms are equal. Evaluating the first term of the derivative is quite simple. However, the sum over all possible labellings in the second term poses more difficulties. This term can be factorised, yielding

$$\sum_t \sum_{y',y} p(Y_{t-1} = y', Y_t = y|\mathbf{x}^{(i)}) f_k(t, y', y, \mathbf{x}^{(i)})$$

This term uses the marginal distribution over pairs of labels, which can be efficiently computed from the forward and backward values as

$$\frac{\alpha_{t-1}(y') \exp \sum_k \lambda_k f_k(t, y', y, \mathbf{x}^{(i)}) \beta_t(y)}{Z(\mathbf{x}^{(i)})}$$

The backward probabilities β are defined by the recursive relation

$$\beta_t(y) = \sum_{y'} \beta_{t+1}(y') \exp \sum_k \lambda_k f_k(t+1, y, y', \mathbf{x})$$

Typically CRF training using LMVM requires many hundreds or thousands of iterations, each of which involves calculating of the log-likelihood and its derivative. The time complexity of a single iteration is $O(L^2NTF)$ where L is the number of labels, N is the number of sequences, T is the average length of the sequences, and F is the average number of activated features of each labelled clique. It is not currently possible to state precise bounds on the number of iterations required for certain problems; however, problems with a large number of sequences often require many more iterations to converge than problems with fewer sequences. Note that efficient CRF implementations cache the feature values for every possible clique labelling of the training data, which leads to a memory requirement with the same complexity of $O(L^2NTF)$ – quite demanding even for current computer hardware.

3 Error Correcting Output Codes

Since the time and space complexity of CRF estimation is dominated by the square of the number of labels, it follows that reducing the number of labels will significantly reduce the complexity. Error-correcting coding is an approach which recasts multiple label problems into a set of binary label problems, each of which is of lesser complexity than the full multiclass problem. Interestingly, training a set of binary CRF classifiers is overall much more efficient than training a full multi-label model. This is because error-correcting CRF training reduces the L^2 complexity term to a constant. Decoding proceeds by predicting these binary labels and then recovering the encoded actual label.

Error-correcting output codes have been used for text classification, as in Berger (1999), on which the following is based. Begin by assigning to each of the m labels a unique n-bit string $\mathcal{C}_i$, which we will call the *code* for this label. Now train n binary classifiers, one for each column of the coding matrix (constructed by taking the labels' *codes* as rows). The j^{th} classifier, γ^j, takes as positive instances those with label i where $\mathcal{C}_{ij} = 1$. In this way, each classifier learns a different concept, discriminating between different subsets of the labels.

We denote the set of binary classifiers as $\Gamma = \{\gamma^1, \gamma^2, \ldots, \gamma^n\}$, which can be used for prediction as follows. Classify a novel instance x with each of the binary classifiers, yielding a n-bit vector $\Gamma(x) = \{\gamma^1(x), \gamma^2(x), \ldots, \gamma^n(x)\}$. Now compare this vector to the codes for each label. The vector may not exactly match any of the labels due to errors in the individual classifiers, and thus we chose the actual label which minimises the distance $\text{argmin}_i \Delta(\Gamma(x), \mathcal{C}_i)$. Typically the Hamming distance is used, which simply measures the number of differing bit positions. In this manner, prediction is resilient to a number of prediction errors by the binary classifiers, provided the codes for the labels are sufficiently diverse.

3.1 Error-correcting CRF training

Error-correcting codes can also be applied to sequence labellers, such as CRFs, which are capable of multiclass labelling. ECOCs can be used with CRFs in a similar manner to that given above for classifiers. A series of CRFs are trained, each on a relabelled variant of the training data. The relabelling for each binary CRF maps the labels into binary space using the relevant column of the coding matrix, such that label i is taken as a positive for the j^{th} model example if $\mathcal{C}_{ij} = 1$.

Training with a binary label set reduces the time and space complexity for each training iteration to $O(NTF)$; the L^2 term is now a constant. Provided the code is relatively short (i.e. there are few binary models, or *weak* learners), this translates into considerable time and space savings. Coding theory doesn't offer any insights into the optimal code length (i.e. the number of weak learners). When using a very short code, the error-correcting CRF will not adequately model the decision boundaries between all classes. However, using a long code will lead to a higher degree of dependency between pairs of classifiers, where both model similar concepts. The generalisation performance should improve quickly as the number of weak learners (code length) increases, but these gains will diminish as the inter-classifier dependence increases.

3.2 Error-correcting CRF decoding

While training of error-correcting CRFs is simply a logical extension of the ECOC classifier method to sequence labellers, decoding is a different matter. We have applied three decoding different strategies. The **Standalone** method requires each binary CRF to find the Viterbi path for a given sequence, yielding a string of 0s and 1s for each model. For each position t in the sequence, the t^{th} bit from each model is taken, and the resultant bit string compared to each of the label codes. The label with the minimum Hamming distance is then chosen as the predicted label for that site. This method allows for error correction to occur at each site, however it discards information about the uncertainty of each weak learner, instead only considering the most probable paths.

The **Marginals** method of decoding uses the marginal probability distribution at each position in the sequence instead of the Viterbi paths. This distribution is easily computed using the forward backward algorithm. The decoding proceeds as before, however instead of a bit string we have a vector of probabilities. This vector is compared to each of the label codes using the L_1 distance, and the closest label is chosen. While this method incorporates the uncertainty of the binary models, it does so at the expense of the path information in the sequence.

Neither of these decoding methods allow the models to interact, although each individual weak learner may benefit from the predictions of the other weak learners. The **Product** decoding method addresses this problem. It treats each weak model as an independent predictor of the label sequence, such that the probability of the label sequence given the observations can be re-expressed as the product of the probabilities assigned by each weak model. A given labelling $\mathbf{y}$ is projected into a bit string for each weak learner, such that the i^{th} entry in the string is $\mathcal{C}_{kj}$ for the j^{th} weak learner, where k is the index of label $\mathbf{y}_i$. The weak learners can then estimate the probability of the bit string; these are then combined into a global product to give the probability of the label sequence

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z'(\mathbf{x})} \prod_j p_j(b_j(\mathbf{y})|\mathbf{x})$$

where $p_j(\mathbf{q}|\mathbf{x})$ is the predicted probability of $\mathbf{q}$ given $\mathbf{x}$ by the j^{th} weak learner, $b_j(\mathbf{y})$ is the bit string representing $\mathbf{y}$ for the j^{th} weak learner and $Z'(\mathbf{x})$ is the partition function. The log probability is

$$\sum_j \{F_j(b_j(\mathbf{y}), \mathbf{x}) \cdot \lambda_\mathbf{j} - \log Z_j(\mathbf{x})\} - \log Z'(\mathbf{x})$$

where $F_j(\mathbf{y}, \mathbf{x}) = \sum_{t=1}^{T+1} \mathbf{f}_j(t, \mathbf{y_{t-1}}, \mathbf{y_t}, \mathbf{x})$. This log probability can then be maximised using the Viterbi algorithm as before, noting that the two log terms are constant with respect to y and thus need not be evaluated. Note that this decoding is an equivalent formulation to a uniformly weighted logarithmic opinion pool, as described in Smith et al. (2005).

Of the three decoding methods, **Standalone** has the lowest complexity, requiring only a binary Viterbi decoding for each weak learner. **Marginals** is slightly more complex, requiring the forward and backward values. **Product**, however, requires Viterbi decoding with the full label set, and many features – the union of the features of each weak learner – which can be quite computationally demanding.

3.3 Choice of code

The accuracy of ECOC methods are highly dependent on the quality of the code. The ideal code has diverse rows, yielding a high error-correcting capability, and diverse columns such that the weak learners model highly independent concepts. When the number of labels, k, is small, an exhaustive code with every unique column is reasonable, given there are $2^{k-1} - 1$ unique columns. With larger label sets, columns must be selected with care to maximise the inter-row and inter-column separation. This can be done by randomly sampling the column space, in which case the probability of poor separation diminishes quickly as the number of columns increases (Berger, 1999). Algebraic codes, such as BCH codes, are an alternative coding scheme which can provide near-optimal error-correcting capability (MacWilliams and Sloane, 1977), however these codes provide no guarantee of good column separation.

4 Experiments

Our experiments show that error-correcting CRFs are highly accurate on benchmark problems with small label sets, as well as on larger problems with many more labels, which would be otherwise prove intractable for traditional CRFs. Moreover, with a good code, the time and resources required for training and decoding can be much less than that of the standardly formulated CRF.

4.1 Named entity recognition

CRFs have been used with strong results on the CoNLL 2003 NER task (McCallum, 2003) and thus this task is included here as a benchmark. This data set consists of a 14,987 training sentences (204,567 tokens) drawn from news articles, tagged for person, location, organisation and miscellaneous entities. There are 8 IOB-2 style labels.

A multiclass (standardly formulated) CRF was trained on these data using features covering word identity, word prefix and suffix, orthographic tests for digits, case and internal punctuation, word length, POS tag and POS tag bigrams before and after the current word. Only features seen at least once in the training data were included in the model, resulting in 450,345 binary features. The model was trained without regularisation and with a Gaussian prior. An exhaustive code was created with all 127 unique columns. All of the weak learners were trained with the same feature set, each having around 315,000 features. The performance of the standard and error-correcting models are shown in Table 1. We tested for statistical significance using the matched pairs test (Gillick and Cox, 1989) at $p < 0.001$. Those results which are significantly better than the corresponding multiclass MLE or regularised model are flagged with a *, and those which are significantly worse with a †.

Model	Decoding	MLE	Regularised
Multiclass		88.04	89.78
Coded	standalone	88.23*	88.67†
	marginals	88.23*	89.19
	product	88.69*	89.69

Table 1: F_1 scores on NER task.

These results show that error-correcting CRF training achieves quite similar performance to the multiclass CRF on the task (which incidentally exceeds McCallum (2003)'s result of 89.0 using feature induction). Product decoding was the better of the three methods, giving the best performance both with and without regularisation, although this difference was only statistically significant between the regularised standalone and the regularised product decoding. The unregularised error-correcting CRF significantly outperformed the multiclass CRF with all decoding strategies, suggesting that the method already provides some regularisation, or corrects some inherent bias in the model.

Using such a large number of weak learners is costly, in this case taking roughly ten times longer to train than the multiclass CRF. However, much shorter codes can also achieve similar results. The simplest code, where each weak learner predicts only a single label (a.k.a. one-vs-all), achieved an F score of 89.56, while only requiring 8 weak learners and less than half the training time as the multiclass CRF. This code has no error correcting capability, suggesting that the code's column separation (and thus interdependence between weak learners) is more important than its row separation.

An exhaustive code was used in this experiment simply for illustrative purposes: many columns in this code were unnecessary, yielding only a slight gain in performance over much simpler codes while incurring a very large increase in training time. Therefore, by selecting a good subset of the exhaustive code, it should be possible to reduce the training time while preserving the strong generalisation performance. One approach is to incorporate skew in the label distribution in our choice of code – the code should minimise the confusability of commonly occurring labels more so than that of rare labels. Assuming that errors made by the weak learners are independent, the probability of a single error, q, as a function of the code length n can be bounded by

$$q(n) \leq 1 - \sum_l p(l) \sum_{i=0}^{\lfloor \frac{h_l - 1}{2} \rfloor} \binom{n}{i} \hat{p}^i (1 - \hat{p})^{n-i}$$

where $p(l)$ is the marginal probability of the label l, h_l is the minimum Hamming distance between l and any other label, and $\hat{p}$ is the maximum probability of an error by a weak learner. The performance achieved by selecting the code with the minimum loss bound from a large random sample of codes is shown in Figure 1, using standalone decoding, where $\hat{p}$ was estimated on the development set. For comparison, randomly sampled codes and a greedy oracle are shown. The two random sampled codes show those samples where no column is repeated, and where duplicate columns are permitted (random with replacement). The oracle repeatedly adds to the code the column which most improves its F_1 score. The minimum loss bound method allows the performance plateau to be reached more quickly than random sampling; i.e. shorter codes can be used, thus allowing more efficient training and decoding.

Note also that multiclass CRF training required 830Mb of memory, while error-correcting training required only 380Mb. Decoding of the test set (51,362 tokens) with the error-correcting model (exhaustive, MLE) took between 150 seconds for standalone decoding and 173 seconds for integrated decoding. The multiclass CRF was much faster, taking only 31 seconds, however this time difference could be reduced with suitable optimisations.

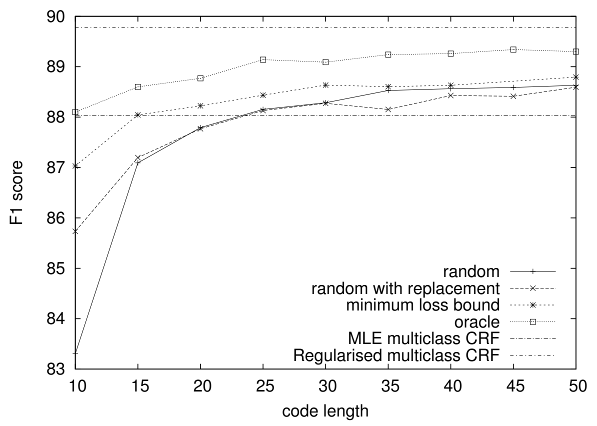


Figure 1: NER F1 scores for standalone decoding with random codes, a minimum loss code and a greedy oracle.

Coding	Decoding	MLE	Regularised
Multiclass		95.69	95.78
Coded - 200	standalone	95.63	96.03
	marginals	95.68	96.03
One-vs-all	product	94.90	96.57

Table 2: POS tagging accuracy.

4.2 Part-of-speech Tagging

CRFs have been applied to POS tagging, however only with a very simple feature set and small training sample (Lafferty et al., 2001). We used the Penn Treebank Wall Street Journal articles, training on sections 2–21 and testing on section 24. In this task there are 45,110 training sentences, a total of 1,023,863 tokens and 45 labels.

The features used included word identity, prefix and suffix, whether the word contains a number, uppercase letter or a hyphen, and the words one and two positions before and after the current word. A random code of 200 columns was used for this task. These results are shown in Table 2, along with those of a multiclass CRF and an alternative one-vs-all coding. As for the NER experiment, the decoding performance levelled off after 100 bits, beyond which the improvements from longer codes were only very slight. This is a very encouraging characteristic, as only a small number of weak learners are required for good performance.

The random code of 200 bits required 1,300Mb of RAM, taking a total of 293 hours to train and 3 hours to decode (54,397 tokens) on similar machines to those used before. We do not have figures regarding the resources used by Lafferty et al.'s CRF for the POS tagging task and our attempts to train a multiclass CRF for full-scale POS tagging were thwarted due to lack of sufficient available computing resources. Instead we trained on a 10,000 sentence subset of the training data, which required approximately 17Gb of RAM and 208 hours to train.

Our best result on the task was achieved using a one-vs-all code, which reduced the training time to 25 hours, as it only required training 45 binary models. This result exceeds Lafferty et al.'s accuracy of 95.73% using a CRF but falls short of Toutanova et al. (2003)'s state-of-the-art 97.24%. This is most probably due to our only using a first-order Markov model and a fairly simple feature set, where Tuotanova et al. include a richer set of features in a third order model.

4.3 Part-of-speech Tagging and Noun Phrase Segmentation

The joint task of simultaneously POS tagging and noun phrase chunking (NPC) was included in order to demonstrate the scalability of error-correcting CRFs. The data was taken from the CoNLL 2000 NPC shared task, with the model predicting both the chunk tags and the POS tags. The training corpus consisted of 8,936 sentences, with 47,377 tokens and 118 labels.

A 200-bit random code was used, with the following features: word identity within a window, prefix and suffix of the current word and the presence of a digit, hyphen or upper case letter in the current word. This resulted in about 420,000 features for each weak learner. A joint tagging accuracy of 90.78% was achieved using MLE training and standalone decoding. Despite the large increase in the number of labels in comparison to the earlier tasks, the performance also began to plateau at around 100 bits. This task required 220Mb of RAM and took a total of 30 minutes to train each of the 200 binary CRFs, this time on Pentium 4 machines with 1Gb RAM. Decoding of the 47,377 test tokens took 9,748 seconds and 9,870 seconds for the standalone and marginals methods respectively.

Sutton et al. (2004) applied a variant of the CRF, the dynamic CRF (DCRF), to the same task, modelling the data with two interconnected chains where one chain predicted NPC tags and the other POS tags. They achieved better performance and training times than our model; however, this is not a fair comparison, as the two approaches are orthogonal. Indeed, applying the error-correcting CRF algorithms to DCRF models could feasibly decrease the complexity of the DCRF, allowing the method to be applied to larger tasks with richer graphical structures and larger label sets.

In all three experiments, error-correcting CRFs have achieved consistently good generalisation performance. The number of weak learners required to achieve these results was shown to be relatively small, even for tasks with large label sets. The time and space requirements were lower than those of a traditional CRF for the larger tasks and, most importantly, did not increase substantially when the number of labels was increased.

5 Related work

Most recent work on improving CRF performance has focused on feature selection. McCallum (2003) describes a technique for greedily adding those feature conjuncts to a CRF which significantly improve the model's log-likelihood. His experimental results show that feature induction yields a large increase in performance, however our results show that standardly formulated CRFs can perform well above their reported 73.3%, casting doubt on the magnitude of the possible improvement. Roark et al. (2004) have also employed feature selection to the huge task of language modelling with a CRF, by partially training a voted perceptron then removing all features that the are ignored by the perceptron. The act of automatic feature selection can be quite time consuming in itself, while the performance and runtime gains are often modest. Even with a reduced number of features, tasks with a very large label space are likely to remain intractable.

6 Conclusion

Standard training methods for CRFs suffer greatly from their dependency on the number of labels, making tasks with large label sets either difficult or impossible. As CRFs are deployed more widely to tasks with larger label sets this problem will become more evident. The current 'solutions' to these scaling problems – namely feature selection, and the use of large clusters – don't address the heart of the problem: the dependence on the square of number of labels.

Error-correcting CRF training allows CRFs to be applied to larger problems and those with larger label sets than were previously possible, without requiring computationally demanding methods such as feature selection. On standard tasks we have shown that error-correcting CRFs provide comparable or better performance than the standardly formulated CRF, while requiring less time and space to train. Only a small number of weak learners were required to obtain good performance on the tasks with large label sets, demonstrating that the method provides efficient scalability to the CRF framework.

Error-correction codes could be applied to other sequence labelling methods, such as the voted perceptron (Roark et al., 2004). This may yield an increase in performance and efficiency of the method, as its runtime is also heavily dependent on the number of labels. We plan to apply error-correcting coding to dynamic CRFs, which should result in better modelling of naturally layered tasks, while increasing the efficiency and scalability of the method. We also plan to develop higher order CRFs, using error-correcting codes to curb the increase in complexity.

7 Acknowledgements

This work was supported in part by a PORES travelling scholarship from the University of Melbourne, allowing Trevor Cohn to travel to Edinburgh.

References

Adam Berger. 1999. Error-correcting output coding for text classification. In *Proceedings of IJCAI: Workshop on machine learning for information filtering*.

Thomas G. Dietterich and Ghulum Bakiri. 1995. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Reseach*, 2:263–286.

L. Gillick and Stephen Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, pages 532–535, Glasgow, Scotland.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In *Proceedings of ICML 2001*, pages 282–289.

Florence MacWilliams and Neil Sloane. 1977. *The theory of error-correcting codes*. North Holland, Amsterdam.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL 2002*, pages 49–55.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL 2003*, pages 188–191.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields. In *Proceedings of UAI 2003*, pages 403–410.

David Pinto, Andrew McCallum, Xing Wei, and Bruce Croft. 2003. Table extraction using conditional random fields. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 235–242.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of ACL 2004*, pages 48–55.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL 2000 and LLL 2000*, pages 127–132.

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.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL 2003*, pages 213–220.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proceedings of ACL 2005*.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labelling and segmenting sequence data. In *Proceedings of the ICML 2004*.

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.

Hanna Wallach. 2002. Efficient training of conditional random fields. Master's thesis, University of Edinburgh.

Logarithmic Opinion Pools for Conditional Random Fields

Andrew Smith
Division of Informatics
University of Edinburgh
United Kingdom
a.p.smith-2@sms.ed.ac.uk

Trevor Cohn
Department of Computer Science
and Software Engineering
University of Melbourne, Australia
tacohn@csse.unimelb.edu.au

Miles Osborne
Division of Informatics
University of Edinburgh
United Kingdom
miles@inf.ed.ac.uk

Abstract

Recent work on Conditional Random Fields (CRFs) has demonstrated the need for regularisation to counter the tendency of these models to overfit. The standard approach to regularising CRFs involves a prior distribution over the model parameters, typically requiring search over a hyperparameter space. In this paper we address the overfitting problem from a different perspective, by factoring the CRF distribution into a weighted product of individual "expert" CRF distributions. We call this model a **logarithmic opinion pool** (LOP) of CRFs (LOP-CRFs). We apply the LOP-CRF to two sequencing tasks. Our results show that unregularised expert CRFs with an unregularised CRF under a LOP can outperform the unregularised CRF, and attain a performance level close to the regularised CRF. LOP-CRFs therefore provide a viable alternative to CRF regularisation without the need for hyperparameter search.

1 Introduction

In recent years, **conditional random fields** (CRFs) (Lafferty et al., 2001) have shown success on a number of natural language processing (NLP) tasks, including shallow parsing (Sha and Pereira, 2003), named entity recognition (McCallum and Li, 2003) and information extraction from research papers (Peng and McCallum, 2004). In general, this work has demonstrated the susceptibility of CRFs to overfit the training data during parameter estimation. As a consequence, it is now standard to use some form of overfitting reduction in CRF training.

Recently, there have been a number of sophisticated approaches to reducing overfitting in CRFs, including automatic feature induction (McCallum, 2003) and a full Bayesian approach to training and inference (Qi et al., 2005). These advanced methods tend to be difficult to implement and are often computationally expensive. Consequently, due to its ease of implementation, the current standard approach to reducing overfitting in CRFs is the use of a prior distribution over the model parameters, typically a Gaussian. The disadvantage with this method, however, is that it requires adjusting the value of one or more of the distribution's hyperparameters. This usually involves manual or automatic tuning on a development set, and can be an expensive process as the CRF must be retrained many times for different hyperparameter values.

In this paper we address the overfitting problem in CRFs from a different perspective. We factor the CRF distribution into a weighted product of individual **expert** CRF distributions, each focusing on a particular subset of the distribution. We call this model a **logarithmic opinion pool** (LOP) of CRFs (LOP-CRFs), and provide a procedure for learning the weight of each expert in the product. The LOP-CRF framework is "parameter-free" in the sense that it does not involve the requirement to adjust hyperparameter values.

LOP-CRFs are theoretically advantageous in that their Kullback-Leibler divergence with a given distribution can be explicitly represented as a function of the KL-divergence with each of their expert distributions. This provides a well-founded framework for designing new overfitting reduction schemes:

Proceedings of the 43rd Annual Meeting of the ACL, pages 18–25,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

look to factorise a CRF distribution as a set of diverse experts.

We apply LOP-CRFs to two sequencing tasks in NLP: named entity recognition and part-of-speech tagging. Our results show that combination of unregularised expert CRFs with an unregularised standard CRF under a LOP can outperform the unregularised standard CRF, and attain a performance level that rivals that of the regularised standard CRF. LOP-CRFs therefore provide a viable alternative to CRF regularisation without the need for hyperparameter search.

2 Conditional Random Fields

A linear chain CRF defines the conditional probability of a state or label sequence **s** given an observed sequence **o** via[1]:

$$p(\mathbf{s} \mid \mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp\left(\sum_{t=1}^{T+1} \sum_k \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t) \right) \quad (1)$$

where T is the length of both sequences, λ_k are parameters of the model and $Z(\mathbf{o})$ is the partition function that ensures (1) represents a probability distribution. The functions f_k are feature functions representing the occurrence of different events in the sequences **s** and **o**.

The parameters λ_k can be estimated by maximising the conditional log-likelihood of a set of labelled training sequences. The log-likelihood is given by:

$$\begin{aligned} \mathscr{L}(\lambda) &= \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \log p(\mathbf{s} \mid \mathbf{o}; \lambda) \\ &= \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \left[\sum_{t=1}^{T+1} \lambda \cdot \mathbf{f}(\mathbf{s}, \mathbf{o}, t) \right] \\ &- \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z(\mathbf{o}; \lambda) \end{aligned}$$

where $\tilde{p}(\mathbf{o},\mathbf{s})$ and $\tilde{p}(\mathbf{o})$ are empirical distributions defined by the training set. At the maximum likelihood solution the model satisfies a set of feature constraints, whereby the expected count of each feature under the model is equal to its empirical count on the training data:

[1]In this paper we assume there is a one-to-one mapping between states and labels, though this need not be the case.

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] = 0, \ \forall k$$

In general this cannot be solved for the λ_k in closed form so numerical routines must be used. Malouf (2002) and Sha and Pereira (2003) show that gradient-based algorithms, particularly limited memory variable metric (LMVM), require much less time to reach convergence, for some NLP tasks, than the iterative scaling methods (Della Pietra et al., 1997) previously used for log-linear optimisation problems. In all our experiments we use the LMVM method to train the CRFs.

For CRFs with general graphical structure, calculation of $E_{p(\mathbf{s}|\mathbf{o})}[f_k]$ is intractable, but for the linear chain case Lafferty et al. (2001) describe an efficient dynamic programming procedure for inference, similar in nature to the forward-backward algorithm in hidden Markov models.

3 Logarithmic Opinion Pools

In this paper an **expert** model refers a probabilistic model that focuses on modelling a specific subset of some probability distribution. The concept of combining the distributions of a set of expert models via a weighted product has previously been used in a range of different application areas, including economics and management science (Bordley, 1982), and NLP (Osborne and Baldridge, 2004).

In this paper we restrict ourselves to sequence models. Given a set of sequence model experts, indexed by α, with conditional distributions $p_\alpha(\mathbf{s} \mid \mathbf{o})$ and a set of non-negative normalised weights w_α, a **logarithmic opinion pool** [2] is defined as the distribution:

$$p_{\text{LOP}}(\mathbf{s} \mid \mathbf{o}) = \frac{1}{Z_{\text{LOP}}(\mathbf{o})} \prod_\alpha [p_\alpha(\mathbf{s} \mid \mathbf{o})]^{w_\alpha} \quad (2)$$

with $w_\alpha \geq 0$ and $\sum_\alpha w_\alpha = 1$, and where $Z_{\text{LOP}}(\mathbf{o})$ is the normalisation constant:

$$Z_{\text{LOP}}(\mathbf{o}) = \sum_{\mathbf{s}} \prod_\alpha [p_\alpha(\mathbf{s} \mid \mathbf{o})]^{w_\alpha} \quad (3)$$

[2]Hinton (1999) introduced a variant of the LOP idea called *Product of Experts*, in which expert distributions are multiplied under a uniform weight distribution.

The weight w_α encodes our confidence in the opinion of expert α.

Suppose that there is a "true" conditional distribution $q(\mathbf{s}\,|\,\mathbf{o})$ which each $p_\alpha(\mathbf{s}\,|\,\mathbf{o})$ is attempting to model. Heskes (1998) shows that the KL divergence between $q(\mathbf{s}\,|\,\mathbf{o})$ and the LOP, can be decomposed into two terms:

$$\begin{aligned} K(q, p_{\text{LOP}}) &= E - A \qquad (4) \\ &= \sum_\alpha w_\alpha K(q, p_\alpha) - \sum_\alpha w_\alpha K(p_{\text{LOP}}, p_\alpha) \end{aligned}$$

This tells us that the closeness of the LOP model to $q(\mathbf{s}\,|\,\mathbf{o})$ is governed by a trade-off between two terms: an E term, which represents the closeness of the individual experts to $q(\mathbf{s}\,|\,\mathbf{o})$, and an A term, which represents the closeness of the individual experts to the LOP, and therefore indirectly to each other. Hence for the LOP to model q well, we desire models p_α which are individually good models of q (having low E) and are also diverse (having large A).

3.1 LOPs for CRFs

Because CRFs are log-linear models, we can see from equation (2) that CRF experts are particularly well suited to combination under a LOP. Indeed, the resulting LOP is itself a CRF, the LOP-CRF, with potential functions given by a log-linear combination of the potential functions of the experts, with weights w_α. As a consequence of this, the normalisation constant for the LOP-CRF can be calculated efficiently via the usual forward-backward algorithm for CRFs. Note that there is a distinction between normalisation constant for the LOP-CRF, Z_{LOP} as given in equation (3), and the partition function of the LOP-CRF, Z. The two are related as follows:

$$\begin{aligned} p_{\text{LOP}}(\mathbf{s}\,|\,\mathbf{o}) &= \frac{1}{Z_{\text{LOP}}(\mathbf{o})} \prod_\alpha [p_\alpha(\mathbf{s}\,|\,\mathbf{o})]^{w_\alpha} \\ &= \frac{1}{Z_{\text{LOP}}(\mathbf{o})} \prod_\alpha \left[\frac{U_\alpha(\mathbf{s}\,|\,\mathbf{o})}{Z_\alpha(\mathbf{o})} \right]^{w_\alpha} \\ &= \frac{\prod_\alpha [U_\alpha(\mathbf{s}\,|\,\mathbf{o})]^{w_\alpha}}{Z_{\text{LOP}}(\mathbf{o}) \prod_\alpha [Z_\alpha(\mathbf{o})]^{w_\alpha}} \end{aligned}$$

where $U_\alpha = \exp \sum_{t=1}^{T+1} \sum_k \lambda_{\alpha k} f_{\alpha k}(s_{t-1}, s_t, \mathbf{o}, t)$ and so

$$\log Z(\mathbf{o}) = \log Z_{\text{LOP}}(\mathbf{o}) + \sum_\alpha w_\alpha \log Z_\alpha(\mathbf{o})$$

This relationship will be useful below, when we describe how to train the weights w_α of a LOP-CRF.

In this paper we will use the term LOP-CRF *weights* to refer to the weights w_α in the weighted product of the LOP-CRF distribution and the term *parameters* to refer to the parameters $\lambda_{\alpha k}$ of each expert CRF α.

3.2 Training LOP-CRFs

In our LOP-CRF training procedure we first train the expert CRFs unregularised on the training data. Then, treating the experts as static pre-trained models, we train the LOP-CRF weights w_α to maximise the log-likelihood of the training data. This training process is "parameter-free" in that neither stage involves the use of a prior distribution over expert CRF parameters or LOP-CRF weights, and so avoids the requirement to adjust hyperparameter values.

The likelihood of a data set under a LOP-CRF, as a function of the LOP-CRF weights, is given by:

$$\begin{aligned} L(\mathbf{w}) &= \prod_{\mathbf{o},\mathbf{s}} p_{\text{LOP}}(\mathbf{s}\,|\,\mathbf{o}; \mathbf{w})^{\tilde{p}(\mathbf{o},\mathbf{s})} \\ &= \prod_{\mathbf{o},\mathbf{s}} \left[\frac{1}{Z_{\text{LOP}}(\mathbf{o}; \mathbf{w})} \prod_\alpha p_\alpha(\mathbf{s}\,|\,\mathbf{o})^{w_\alpha} \right]^{\tilde{p}(\mathbf{o},\mathbf{s})} \end{aligned}$$

After taking logs and rearranging, the log-likelihood can be expressed as:

$$\begin{aligned} \mathscr{L}(\mathbf{w}) &= \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \sum_\alpha w_\alpha \log p_\alpha(\mathbf{s}\,|\,\mathbf{o}) \\ &- \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z_{\text{LOP}}(\mathbf{o}; \mathbf{w}) \\ &= \sum_\alpha w_\alpha \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \log p_\alpha(\mathbf{s}\,|\,\mathbf{o}) \\ &+ \sum_\alpha w_\alpha \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z_\alpha(\mathbf{o}) \\ &- \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z(\mathbf{o}; \mathbf{w}) \end{aligned}$$

For the first two terms, the quantities that are multiplied by w_α inside the (outer) sums are independent of the weights, and can be evaluated once at the beginning of training. The third term involves the partition function for the LOP-CRF and so is a function of the weights. It can be evaluated efficiently as usual for a standard CRF.

Taking derivatives with respect to w_β and rearranging, we obtain:

$$\begin{aligned}\frac{\partial \mathscr{L}(\mathbf{w})}{\partial w_\beta} &= \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \log p_\beta(\mathbf{s}\,|\,\mathbf{o}) \\ &+ \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z_\beta(\mathbf{o}) \\ &- \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) E_{p_{\text{LOP}}(\mathbf{s}|\mathbf{o})}\left[\sum_t \log U_{\beta t}(\mathbf{o},\mathbf{s})\right]\end{aligned}$$

where $U_{\beta t}(\mathbf{o},\mathbf{s})$ is the value of the potential function for expert β on clique t under the labelling $\mathbf{s}$ for observation $\mathbf{o}$. In a way similar to the representation of the expected feature count in a standard CRF, the third term may be re-written as:

$$-\sum_{\mathbf{o}}\sum_t\sum_{s',s''} p_{\text{LOP}}(s_{t-1}=s', s_t=s'', \mathbf{o}) \log U_{\beta t}(s',s'',\mathbf{o})$$

Hence the derivative is tractable because we can use dynamic programming to efficiently calculate the pairwise marginal distribution for the LOP-CRF.

Using these expressions we can efficiently train the LOP-CRF weights to maximise the log-likelihood of the data set.[3] We make use of the LMVM method mentioned earlier to do this. We will refer to a LOP-CRF with weights trained using this procedure as an *unregularised* LOP-CRF.

3.2.1 Regularisation

The "parameter-free" aspect of the training procedure we introduced in the previous section relies on the fact that we do not use regularisation when training the LOP-CRF weights w_α. However, there is a possibility that this may lead to overfitting of the training data. In order to investigate this, we develop a regularised version of the training procedure and compare the results obtained with each. We use a prior distribution over the LOP-CRF weights. As the weights are non-negative and normalised we use a Dirichlet distribution, whose density function is given by:

$$p(\mathbf{w}) = \frac{\Gamma(\sum_\alpha \theta_\alpha)}{\prod_\alpha \Gamma(\theta_\alpha)} \prod_\alpha w_\alpha^{\theta_\alpha - 1}$$

where the θ_α are hyperparameters.

Under this distribution, ignoring terms that are independent of the weights, the regularised log-likelihood involves an additional term:

$$\sum_\alpha (\theta_\alpha - 1) \log w_\alpha$$

We assume a single value θ across all weights. The derivative of the regularised log-likelihood with respect to weight w_β then involves an additional term $\frac{1}{w_\beta}(\theta - 1)$. In our experiments we use the development set to optimise the value of θ. We will refer to a LOP-CRF with weights trained using this procedure as a *regularised* LOP-CRF.

4 The Tasks

In this paper we apply LOP-CRFs to two sequence labelling tasks in NLP: **named entity recognition** (NER) and **part-of-speech tagging** (POS tagging).

4.1 Named Entity Recognition

NER involves the identification of the location and type of pre-defined entities within a sentence and is often used as a sub-process in information extraction systems. With NER the CRF is presented with a set of sentences and must label each word so as to indicate whether the word appears outside an entity (O), at the beginning of an entity of type X (B-X) or within the continuation of an entity of type X (I-X).

All our results for NER are reported on the CoNLL-2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003). For this dataset the entity types are: persons (PER), locations (LOC), organisations (ORG) and miscellaneous (MISC). The training set consists of $14,987$ sentences and $204,567$ tokens, the development set consists of $3,466$ sentences and $51,578$ tokens and the test set consists of $3,684$ sentences and $46,666$ tokens.

[3]We must ensure that the weights are non-negative and normalised. We achieve this by parameterising the weights as functions of a set of unconstrained variables via a softmax transformation. The values of the log-likelihood and its derivatives with respect to the unconstrained variables can be derived from the corresponding values for the weights w_α.

4.2 Part-of-Speech Tagging

POS tagging involves labelling each word in a sentence with its part-of-speech, for example noun, verb, adjective, etc. For our experiments we use the CoNLL-2000 shared task dataset (Tjong Kim Sang and Buchholz, 2000). This has 48 different POS tags. In order to make training time manageable[4], we collapse the number of POS tags from 48 to 5 following the procedure used in (McCallum et al., 2003). In summary:

- All types of noun collapse to category **N**.
- All types of verb collapse to category **V**.
- All types of adjective collapse to category **J**.
- All types of adverb collapse to category **R**.
- All other POS tags collapse to category **O**.

The training set consists of $7,300$ sentences and $173,542$ tokens, the development set consists of $1,636$ sentences and $38,185$ tokens and the test set consists of $2,012$ sentences and $47,377$ tokens.

4.3 Expert sets

For each task we compare the performance of the LOP-CRF to that of the standard CRF by defining a single, complex CRF, which we call a **monolithic** CRF, and a range of **expert sets**.

The monolithic CRF for NER comprises a number of word and POS tag features in a window of five words around the current word, along with a set of orthographic features defined on the current word. These are based on those found in (Curran and Clark, 2003). Examples include whether the current word is capitalised, is an initial, contains a digit, contains punctuation, etc. The monolithic CRF for NER has $450,345$ features.

The monolithic CRF for POS tagging comprises word and POS features similar to those in the NER monolithic model, but over a smaller number of orthographic features. The monolithic model for POS tagging has $188,448$ features.

Each of our expert sets consists of a number of CRF experts. Usually these experts are designed to focus on modelling a particular aspect or subset of the distribution. As we saw earlier, the aim here is to define experts that model parts of the distribution well while retaining mutual diversity. The experts from a particular expert set are combined under a LOP-CRF and the weights are trained as described previously.

We define our range of expert sets as follows:

- **Simple** consists of the monolithic CRF and a single expert comprising a reduced subset of the features in the monolithic CRF. This reduced CRF models the entire distribution rather than focusing on a particular aspect or subset, but is much less expressive than the monolithic model. The reduced model comprises $24,818$ features for NER and $47,420$ features for POS tagging.
- **Positional** consists of the monolithic CRF and a partition of the features in the monolithic CRF into three experts, each consisting only of features that involve events either behind, at or ahead of the current sequence position.
- **Label** consists of the monolithic CRF and a partition of the features in the monolithic CRF into five experts, one for each label. For NER an expert corresponding to label X consists only of features that involve labels B-X or I-X at the current or previous positions, while for POS tagging an expert corresponding to label X consists only of features that involve label X at the current or previous positions. These experts therefore focus on trying to model the distribution of a particular label.
- **Random** consists of the monolithic CRF and a random partition of the features in the monolithic CRF into four experts. This acts as a baseline to ascertain the performance that can be expected from an expert set that is not defined via any linguistic intuition.

5 Experiments

To compare the performance of LOP-CRFs trained using the procedure we described previously to that of a standard CRF regularised with a Gaussian prior, we do the following for both NER and POS tagging:

[4]See (Cohn et al., 2005) for a scaling method allowing the full POS tagging task with CRFs.

- Train a monolithic CRF with regularisation using a Gaussian prior. We use the development set to optimise the value of the variance hyperparameter.

- Train every expert CRF in each expert set without regularisation (each expert set includes the monolithic CRF, which clearly need only be trained once).

- For each expert set, create a LOP-CRF from the expert CRFs and train the weights of the LOP-CRF without regularisation. We compare its performance to that of the unregularised and regularised monolithic CRFs.

- To investigate whether training the LOP-CRF weights contributes significantly to the LOP-CRF's performance, for each expert set we create a LOP-CRF with uniform weights and compare its performance to that of the LOP-CRF with trained weights.

- To investigate whether unregularised training of the LOP-CRF weights leads to overfitting, for each expert set we train the weights of the LOP-CRF with regularisation using a Dirichlet prior. We optimise the hyperparameter in the Dirichlet distribution on the development set. We then compare the performance of the LOP-CRF with regularised weights to that of the LOP-CRF with unregularised weights.

6 Results

6.1 Experts

Before presenting results for the LOP-CRFs, we briefly give performance figures for the monolithic CRFs and expert CRFs in isolation. For illustration, we do this for NER models only. Table 1 shows F scores on the development set for the NER CRFs. We see that, as expected, the expert CRFs in isolation model the data relatively poorly compared to the monolithic CRFs. Some of the label experts, for example, attain relatively low F scores as they focus only on modelling one particular label. Similar behaviour was observed for the POS tagging models.

Expert	F score
Monolithic unreg.	88.33
Monolithic reg.	89.84
Reduced	79.62
Positional 1	86.96
Positional 2	73.11
Positional 3	73.08
Label LOC	41.96
Label MISC	22.03
Label ORG	29.13
Label PER	40.49
Label O	60.44
Random 1	70.34
Random 2	67.76
Random 3	67.97
Random 4	70.17

Table 1: Development set F scores for NER experts

6.2 LOP-CRFs with unregularised weights

In this section we present results for LOP-CRFs with unregularised weights. Table 2 gives F scores for NER LOP-CRFs while Table 3 gives accuracies for the POS tagging LOP-CRFs. The monolithic CRF scores are included for comparison. Both tables illustrate the following points:

- In every case the LOP-CRFs outperform the unregularised monolithic CRF

- In most cases the performance of LOP-CRFs rivals that of the regularised monolithic CRF, and in some cases exceeds it.

We use McNemar's matched-pairs test (Gillick and Cox, 1989) on point-wise labelling errors to examine the statistical significance of these results. We test significance at the 5% level. At this threshold, all the LOP-CRFs significantly outperform the corresponding unregularised monolithic CRF. In addition, those marked with * show a significant improvement over the regularised monolithic CRF. Only the value marked with † in Table 3 significantly under performs the regularised monolithic. All other values a do not differ significantly from those of the regularised monolithic CRF at the 5% level.

These results show that LOP-CRFs with unregularised weights can lead to performance improvements that equal or exceed those achieved from a conventional regularisation approach using a Gaussian prior. The important difference, however, is that the LOP-CRF approach is "parameter-free" in the

Expert set	Development set	Test set
Monolithic unreg.	88.33	81.87
Monolithic reg.	89.84	83.98
Simple	90.26	84.22*
Positional	90.35	84.71*
Label	89.30	83.27
Random	88.84	83.06

Table 2: F scores for NER unregularised LOP-CRFs

Expert set	Development set	Test set
Monolithic unreg.	97.92	97.65
Monolithic reg.	98.02	97.84
Simple	98.31*	98.12*
Positional	98.03	97.81
Label	97.99	97.77
Random	97.99	97.76†

Table 3: Accuracies for POS tagging unregularised LOP-CRFs

sense that each expert CRF in the LOP-CRF is unregularised and the LOP weight training is also unregularised. We are therefore not required to search a hyperparameter space. As an illustration, to obtain our best results for the POS tagging regularised monolithic model, we re-trained using 15 different values of the Gaussian prior variance. With the LOP-CRF we trained each expert CRF and the LOP weights only *once*.

As an illustration of a typical weight distribution resulting from the training procedure, the **positional** LOP-CRF for POS tagging attaches weight 0.45 to the monolithic model and roughly equal weights to the other three experts.

6.3 LOP-CRFs with uniform weights

By training LOP-CRF weights using the procedure we introduce in this paper, we allow the weights to take on non-uniform values. This corresponds to letting the opinion of some experts take precedence over others in the LOP-CRF's decision making. An alternative, simpler, approach would be to combine the experts under a LOP with uniform weights, thereby avoiding the weight training stage. We would like to ascertain whether this approach will significantly reduce the LOP-CRF's performance. As an illustration, Table 4 gives accuracies for LOP-CRFs with uniform weights for POS tagging. A similar pattern is observed for NER. Comparing these values to those in Tables 2 and 3, we can see that in

Expert set	Development set	Test set
Simple	98.30	98.12
Positional	97.97	97.79
Label	97.85	97.73
Random	97.82	97.74

Table 4: Accuracies for POS tagging uniform LOP-CRFs

general LOP-CRFs with uniform weights, although still performing significantly better than the unregularised monolithic CRF, generally under perform LOP-CRFs with trained weights. This suggests that the choice of weights can be important, and justifies the weight training stage.

6.4 LOP-CRFs with regularised weights

To investigate whether unregularised training of the LOP-CRF weights leads to overfitting, we train the LOP-CRF with regularisation using a Dirichlet prior. The results we obtain show that in most cases a LOP-CRF with regularised weights achieves an almost identical performance to that with unregularised weights, and suggests there is little to be gained by weight regularisation. This is probably due to the fact that in our LOP-CRFs the number of experts, and therefore weights, is generally small and so there is little capacity for overfitting. We conjecture that although other choices of expert set may comprise many more experts than in our examples, the numbers are likely to be relatively small in comparison to, for example, the number of parameters in the individual experts. We therefore suggest that any overfitting effect is likely to be limited.

6.5 Choice of Expert Sets

We can see from Tables 2 and 3 that the performance of a LOP-CRF varies with the choice of expert set. For example, in our tasks the **simple** and **positional** expert sets perform better than those for the **label** and **random** sets. For an explanation here, we refer back to our discussion of equation (5). We conjecture that the **simple** and **positional** expert sets achieve good performance in the LOP-CRF because they consist of experts that are diverse while simultaneously being reasonable models of the data. The **label** expert set exhibits greater diversity between the experts, because each expert focuses on modelling a particular label only, but each expert is a relatively poor model of the entire distribution and the corresponding LOP-CRF performs worse. Similarly, the **random** experts are in general better models of the entire distribution but tend to be less diverse because they do not focus on any one aspect or subset of it. Intuitively, then, we want to devise experts that provide diverse but accurate views on the data.

The expert sets we present in this paper were motivated by linguistic intuition, but clearly many choices exist. It remains an important open question as to how to automatically construct expert sets for good performance on a given task, and we intend to pursue this avenue in future research.

7 Conclusion and future work

In this paper we have introduced the logarithmic opinion pool of CRFs as a way to address overfitting in CRF models. Our results show that a LOP-CRF can provide a competitive alternative to conventional regularisation with a prior while avoiding the requirement to search a hyperparameter space.

We have seen that, for a variety of types of expert, combination of expert CRFs with an unregularised standard CRF under a LOP with optimised weights can outperform the unregularised standard CRF and rival the performance of a regularised standard CRF.

We have shown how these advantages a LOP-CRF provides have a firm theoretical foundation in terms of the decomposition of the KL-divergence between a LOP-CRF and a target distribution, and how this provides a framework for designing new overfitting reduction schemes in terms of constructing diverse experts.

In this work we have considered training the weights of a LOP-CRF using pre-trained, static experts. In future we intend to investigate cooperative training of LOP-CRF weights and the parameters of each expert in an expert set.

Acknowledgements

We wish to thank Stephen Clark, our colleagues in Edinburgh and the anonymous reviewers for many useful comments.

References

R. F. Bordley. 1982. A multiplicative formula for aggregating probability assessments. *Management Science*, (28):1137–1148.

T. Cohn, A. Smith, and M. Osborne. 2005. Scaling conditional random fields using error-correcting codes. In *Proc. ACL 2005*.

J. Curran and S. Clark. 2003. Language independent NER using a maximum entropy tagger. In *Proc. CoNLL-2003*.

S. Della Pietra, Della Pietra V., and J. Lafferty. 1997. Inducing features of random fields. In *IEEE PAMI*, volume 19(4), pages 380–393.

L. Gillick and S. Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 532–535.

T. Heskes. 1998. Selecting weighting factors in logarithmic opinion pools. In *Advances in Neural Information Processing Systems 10*.

G. E. Hinton. 1999. Product of experts. In *ICANN*, volume 1, pages 1–6.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML 2001*.

R. Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. CoNLL-2002*.

A. McCallum and W. Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proc. CoNLL-2003*.

A. McCallum, K. Rohanimanesh, and C. Sutton. 2003. Dynamic conditional random fields for jointly labeling multiple sequences. In *NIPS-2003 Workshop on Syntax, Semantics and Statistics*.

A. McCallum. 2003. Efficiently inducing features of conditional random fields. In *Proc. UAI 2003*.

M. Osborne and J. Baldridge. 2004. Ensemble-based active learning for parse selection. In *Proc. NAACL 2004*.

F. Peng and A. McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Proc. HLT-NAACL 2004*.

Y. Qi, M. Szummer, and T. P. Minka. 2005. Bayesian conditional random fields. In *Proc. AISTATS 2005*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. HLT-NAACL 2003*.

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. CoNLL-2000*.

E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. CoNLL-2003*.

Supersense Tagging of Unknown Nouns using Semantic Similarity

James R. Curran
School of Information Technologies
University of Sydney
NSW 2006, Australia
james@it.usyd.edu.au

Abstract

The limited coverage of lexical-semantic resources is a significant problem for NLP systems which can be alleviated by automatically classifying the unknown words. *Supersense tagging* assigns unknown nouns one of 26 broad semantic categories used by lexicographers to organise their manual insertion into WORDNET. Ciaramita and Johnson (2003) present a tagger which uses synonym set glosses as annotated training examples. We describe an unsupervised approach, based on vector-space similarity, which does not require annotated examples but significantly outperforms their tagger. We also demonstrate the use of an extremely large shallow-parsed corpus for calculating vector-space semantic similarity.

1 Introduction

Lexical-semantic resources have been applied successful to a wide range of Natural Language Processing (NLP) problems ranging from collocation extraction (Pearce, 2001) and class-based smoothing (Clark and Weir, 2002), to text classification (Baker and McCallum, 1998) and question answering (Pasca and Harabagiu, 2001). In particular, WORDNET (Fellbaum, 1998) has significantly influenced research in NLP.

Unfortunately, these resource are extremely time-consuming and labour-intensive to manually develop and maintain, requiring considerable linguistic and domain expertise. Lexicographers cannot possibly keep pace with language evolution: sense distinctions are continually made and merged, words are coined or become obsolete, and technical terms migrate into the vernacular. Technical domains, such as medicine, require separate treatment since common words often take on special meanings, and a significant proportion of their vocabulary does not overlap with everyday vocabulary. Burgun and Bodenreider (2001) compared an alignment of WORDNET with the UMLS medical resource and found only a very small degree of overlap. Also, lexical-semantic resources suffer from:

bias towards concepts and senses from particular topics. Some specialist topics are better covered in WORDNET than others, e.g. dog has finer-grained distinctions than cat and worm although this does not reflect finer distinctions in reality;

limited coverage of infrequent words and senses. Ciaramita and Johnson (2003) found that common nouns missing from WORDNET 1.6 occurred every 8 sentences in the BLLIP corpus. By WORDNET 2.0, coverage has improved but the problem of keeping up with language evolution remains difficult.

consistency when classifying similar words into categories. For instance, the WORDNET lexicographer file for ionosphere (location) is different to exosphere and stratosphere (object), two other layers of the earth's atmosphere.

These problems demonstrate the need for automatic or semi-automatic methods for the creation and maintenance of lexical-semantic resources. Broad semantic classification is currently used by lexicographers to organise the manual insertion of words into WORDNET, and is an experimental precursor to automatically inserting words directly into the WORDNET hierarchy. Ciaramita and Johnson (2003) call this *supersense tagging* and describe a multi-class perceptron tagger, which uses WORDNET's hierarchical structure to create many annotated training instances from the synset glosses.

This paper describes an unsupervised approach to supersense tagging that does not require annotated sentences. Instead, we use vector-space similarity to retrieve a number of synonyms for each unknown common noun. The supersenses of these synonyms are then combined to determine the supersense. This approach significantly outperforms the multi-class perceptron on the same dataset based on WORDNET 1.6 and 1.7.1.

LEX-FILE	DESCRIPTION
act	acts or actions
animal	animals
artifact	man-made objects
attribute	attributes of people and objects
body	body parts
cognition	cognitive processes and contents
communication	communicative processes and contents
event	natural events
feeling	feelings and emotions
food	foods and drinks
group	groupings of people or objects
location	spatial position
motive	goals
object	natural objects (not man-made)
person	people
phenomenon	natural phenomena
plant	plants
possession	possession and transfer of possession
process	natural processes
quantity	quantities and units of measure
relation	relations between people/things/ideas
shape	two and three dimensional shapes
state	stable states of affairs
substance	substances
time	time and temporal relations

Table 1: 25 noun lexicographer files in WORDNET

2 Supersenses

There are 26 broad semantic classes employed by lexicographers in the initial phase of inserting words into the WORDNET hierarchy, called *lexicographer files* (*lex-files*). For the noun hierarchy, there are 25 lex-files and a file containing the top level nodes in the hierarchy called Tops. Other syntactic classes are also organised using lex-files: 15 for verbs, 3 for adjectives and 1 for adverbs.

Lex-files form a set of coarse-grained sense distinctions within WORDNET. For example, company appears in the following lex-files in WORDNET 2.0: group, which covers company in the social, commercial and troupe fine-grained senses; and state, which covers companionship. The names and descriptions of the noun lex-files are shown in Table 1. Some lex-files map directly to the top level nodes in the hierarchy, called *unique beginners*, while others are grouped together as hyponyms of a unique beginner (Fellbaum, 1998, page 30). For example, abstraction subsumes the lex-files attribute, quantity, relation, communication and time.

Ciaramita and Johnson (2003) call the noun lex-file classes *supersenses*. There are 11 unique beginners in the WORDNET noun hierarchy which could also be used as supersenses. Ciaramita (2002) has produced a mini-WORDNET by manually reducing the WORDNET hierarchy to 106 broad categories. Ciaramita et al. (2003) describe how the lex-files can be used as root nodes in a two level hierarchy with the WORDNET synsets appearing directly underneath.

Other alternative sets of supersenses can be created by an arbitrary cut through the WORDNET hierarchy near the top, or by using topics from a thesaurus such as Roget's (Yarowsky, 1992). These topic distinctions are coarser-grained than WORDNET senses, which have been criticised for being too difficult to distinguish even for experts. Ciaramita and Johnson (2003) believe that the key sense distinctions are still maintained by supersenses. They suggest that supersense tagging is similar to named entity recognition, which also has a very small set of categories with similar granularity (e.g. location and person) for labelling predominantly unseen terms.

Supersense tagging can provide automated or semi-automated assistance to lexicographers adding words to the WORDNET hierarchy. Once this task is solved successfully, it may be possible to insert words directly into the fine-grained distinctions of the hierarchy itself. Clearly, this is the ultimate goal, to be able to insert new terms into lexical resources, extending the structure where necessary. Supersense tagging is also interesting for many applications that use shallow semantics, e.g. information extraction and question answering.

3 Previous Work

A considerable amount of research addresses structurally and statistically manipulating the hierarchy of WORDNET and the construction of new wordnets using the concept structure from English. For *lexical FreeNet*, Beeferman (1998) adds over 350 000 collocation pairs (*trigger pairs*) extracted from a 160 million word corpus of broadcast news using mutual information. The co-occurrence window was 500 words which was designed to approximate average document length.

Caraballo and Charniak (1999) have explored determining noun specificity from raw text. They find that simple frequency counts are the most effective way of determining the parent-child ordering, achieving 83% accuracy over types of vehicle, food and occupation. The other measure they found to be successful was the entropy of the conditional distribution of surrounding words given the noun. Specificity ordering is a necessary step for building a noun hierarchy. However, this approach clearly cannot build a hierarchy alone. For instance, entity is less frequent than many concepts it subsumes. This suggests it will only be possible to add words to an existing abstract structure rather than create categories right up to the unique beginners.

Hearst and Schütze (1993) flatten WORDNET into 726 categories using an algorithm which attempts to minimise the variance in category size. These categories are used to label paragraphs with topics, effectively repeating Yarowsky's (1992) experiments using the their categories rather than Roget's thesaurus. Schütze's (1992)

WordSpace system was used to add topical links, such as between ball, racquet and game (the *tennis problem*). Further, they also use the same vector-space techniques to label previously unseen words using the most common class assigned to the top 20 synonyms for that word.

Widdows (2003) uses a similar technique to insert words into the WORDNET hierarchy. He first extracts synonyms for the unknown word using vector-space similarity measures based on Latent Semantic Analysis and then searches for a location in the hierarchy nearest to these synonyms. This same technique as is used in our approach to supersense tagging.

Ciaramita and Johnson (2003) implement a supersense tagger based on the multi-class perceptron classifier (Crammer and Singer, 2001), which uses the standard collocation, spelling and syntactic features common in WSD and named entity recognition systems. Their insight was to use the WORDNET glosses as annotated training data and massively increase the number of training instances using the noun hierarchy. They developed an efficient algorithm for estimating the model over hierarchical training data.

4 Evaluation

Ciaramita and Johnson (2003) propose a very natural evaluation for supersense tagging: inserting the extra common nouns that have been added to a new version of WORDNET. They use the common nouns that have been added to WORDNET 1.7.1 since WORDNET 1.6 and compare this evaluation with a standard cross-validation approach that uses a small percentage of the words from their WORDNET 1.6 training set for evaluation. Their results suggest that the WORDNET 1.7.1 test set is significantly harder because of the large number of abstract category nouns, e.g. communication and cognition, that appear in the 1.7.1 data, which are difficult to classify.

Our evaluation will use exactly the same test sets as Ciaramita and Johnson (2003). The WORDNET 1.7.1 test set consists of 744 previously unseen nouns, the majority of which (over 90%) have only one sense. The WORDNET 1.6 test set consists of several cross-validation sets of 755 nouns randomly selected from the BLLIP training set used by Ciaramita and Johnson (2003). They have kindly supplied us with the WORDNET 1.7.1 test set and one cross-validation run of the WORDNET 1.6 test set. Our development experiments are performed on the WORDNET 1.6 test set with one final run on the WORDNET 1.7.1 test set. Some examples from the test sets are given in Table 2 with their supersenses.

5 Corpus

We have developed a 2 billion word corpus, shallow-parsed with a statistical NLP pipeline, which is by far the largest NLP processed corpus described in published research. The corpus consists of the *British National Corpus* (BNC), the *Reuters Corpus Volume 1* (RCV1), and most of the Linguistic Data Consortium's news text collected since 1987: *Continuous Speech Recognition III* (CSR-III); *North American News Text Corpus* (NANTC); the NANTC *Supplement* (NANTS); and the ACQUAINT *Corpus*. The components and their sizes including punctuation are given in Table 3. The LDC has recently released the *English Gigaword* corpus which includes most of the corpora listed above.

WORDNET 1.6		WORDNET 1.7.1	
NOUN	SUPERSENSE	NOUN	SUPERSENSE
stock index	communication	week	time
fast food	food	buyout	act
bottler	group	insurer	group
subcompact	artifact	partner	person
advancer	person	health	state
cash flow	possession	income	possession
downside	cognition	contender	person
discounter	artifact	cartel	group
trade-off	act	lender	person
billionaire	person	planner	artifact

Table 2: Example nouns and their supersenses

CORPUS	DOCS.	SENTS.	WORDS
BNC	4 124	6.2M	114M
RCV1	806 791	8.1M	207M
CSR-III	491 349	9.3M	226M
NANTC	930 367	23.2M	559M
NANTS	942 167	25.2M	507M
ACQUAINT	1 033 461	21.3M	491M

Table 3: 2 billion word corpus statistics

We have tokenized the text using the Grok-OpenNLP tokenizer (Morton, 2002) and split the sentences using MXTerminator (Reynar and Ratnaparkhi, 1997). Any sentences less than 3 words or more than 100 words long were rejected, along with sentences containing more than 5 numbers or more than 4 brackets, to reduce noise. The rest of the pipeline is described in the next section.

6 Semantic Similarity

Vector-space models of similarity are based on the *distributional hypothesis* that similar words appear in similar contexts. This hypothesis suggests that semantic similarity can be measured by comparing the contexts each word appears in. In vector-space models each *headword* is represented by a vector of frequency counts recording the contexts that it appears in. The key parameters are the context extraction method and the similarity measure used to compare context vectors. Our approach to vector-space similarity is based on the SEXTANT system described in Grefenstette (1994).

Curran and Moens (2002b) compared several context extraction methods and found that the shallow pipeline and grammatical relation extraction used in SEXTANT was both extremely fast and produced high-quality results. SEXTANT extracts relation tuples $(\mathrm{w}, r, \mathrm{w}')$ for each noun, where w is the headword, r is the relation type and w' is the other word. The efficiency of the SEXTANT approach makes the extraction of contextual information from over 2 billion words of raw text feasible. We describe the shallow pipeline in detail below.

Curran and Moens (2002a) compared several different similarity measures and found that Grefenstette's weighted JACCARD measure performed the best:

$$\frac{\sum \min(\mathrm{wgt}(w_1, *_r, *_{w'}), \mathrm{wgt}(w_2, *_r, *_{w'}))}{\sum \max(\mathrm{wgt}(w_1, *_r, *_{w'}), \mathrm{wgt}(w_2, *_r, *_{w'}))} \quad (1)$$

where $\mathrm{wgt}(w, r, w')$ is the weight function for relation (w, r, w'). Curran and Moens (2002a) introduced the TTEST weight function, which is used in collocation extraction. Here, the t-test compares the joint and product probability distributions of the headword and context:

$$\frac{p(w, r, w') - p(*, r, w')p(w, *, *)}{\sqrt{p(*, r, w')p(w, *, *)}} \quad (2)$$

where $*$ indicates a global sum over that element of the relation tuple. JACCARD and TTEST produced better quality synonyms than existing measures in the literature, so we use Curran and Moen's configuration for our supersense tagging experiments.

6.1 Part of Speech Tagging and Chunking

Our implementation of SEXTANT uses a maximum entropy POS tagger designed to be very efficient, tagging at around 100 000 words per second (Curran and Clark, 2003), trained on the entire Penn Treebank (Marcus et al., 1994). The only similar performing tool is the *Trigrams 'n' Tags* tagger (Brants, 2000) which uses a much simpler statistical model. Our implementation uses a maximum entropy chunker which has similar feature types to Koeling (2000) and is also trained on chunks extracted from the entire Penn Treebank using the CoNLL 2000 script. Since the Penn Treebank separates PPs and conjunctions from NPs, they are concatenated to match Grefenstette's table-based results, i.e. the SEXTANT always prefers noun attachment.

6.2 Morphological Analysis

Our implementation uses `morpha`, the Sussex morphological analyser (Minnen et al., 2001), which is implemented using `lex` grammars for both affix splitting and generation. `morpha` has wide coverage – nearly 100% against the CELEX lexical database (Minnen et al., 2001) – and is very efficient, analysing over 80 000 words per second. `morpha` often maintains sense distinctions between singular and plural nouns; for instance: spectacles is not reduced to spectacle, but fails to do so in other cases: glasses is converted to glass. This inconsistency is problematic when using morphological analysis to smooth vector-space models. However, morphological smoothing still produces better results in practice.

RELATION	DESCRIPTION
adj	noun–adjectival modifier relation
dobj	verb–direct object relation
iobj	verb–indirect object relation
nn	noun–noun modifier relation
nnprep	noun–prepositional head relation
subj	verb–subject relation

Table 4: Grammatical relations from SEXTANT

6.3 Grammatical Relation Extraction

After the raw text has been POS tagged and chunked, the grammatical relation extraction algorithm is run over the chunks. This consists of five passes over each sentence that first identify noun and verb phrase heads and then collect grammatical relations between each common noun and its modifiers and verbs. A global list of grammatical relations generated by each pass is maintained across the passes. The global list is used to determine if a word is already attached. Once all five passes have been completed this association list contains all of the noun-modifier/verb pairs which have been extracted from the sentence. The types of grammatical relation extracted by SEXTANT are shown in Table 4. For relations between nouns (nn and nnprep), we also create inverse relations (w', r', w) representing the fact that w' can modify w. The 5 passes are described below.

Pass 1: Noun Pre-modifiers

This pass scans NPs, left to right, creating adjectival (adj) and nominal (nn) pre-modifier grammatical relations (GRs) with every noun to the pre-modifier's right, up to a preposition or the phrase end. This corresponds to assuming right-branching noun compounds. Within each NP only the NP and PP heads remain unattached.

Pass 2: Noun Post-modifiers

This pass scans NPs, right to left, creating post-modifier GRs between the unattached heads of NPs and PPs. If a preposition is encountered between the noun heads, a prepositional noun (nnprep) GR is created, otherwise an appositional noun (nn) GR is created. This corresponds to assuming right-branching PP attachment. After this phrase only the NP head remains unattached.

Tense Determination

The rightmost verb in each VP is considered the head. A

VP is initially categorised as active. If the head verb is a form of *be* then the VP becomes attributive. Otherwise, the algorithm scans the VP from right to left: if an auxiliary verb form of be is encountered the VP becomes passive; if a progressive verb (except being) is encountered the VP becomes active.

Only the noun heads on either side of VPs remain unattached. The remaining three passes attach these to the verb heads as either subjects or objects depending on the voice of the VP.

Pass 3: Verb Pre-Attachment

This pass scans sentences, right to left, associating the first NP head to the left of the VP with its head. If the VP is active, a subject (subj) relation is created; otherwise, a direct object (dobj) relation is created. For example, antigen is the subject of represent.

Pass 4: Verb Post-Attachment

This pass scans sentences, left to right, associating the first NP or PP head to the right of the VP with its head. If the VP was classed as active and the phrase is an NP then a direct object (dobj) relation is created. If the VP was classed as passive and the phrase is an NP then a subject (subj) relation is created. If the following phrase is a PP then an indirect object (iobj) relation is created. The interaction between the head verb and the preposition determine whether the noun is an indirect object of a ditransitive verb or alternatively the head of a PP that is modifying the verb. However, SEXTANT always attaches the PP to the previous phrase.

Pass 5: Verb Progressive Participles

The final step of the process is to attach progressive verbs to subjects and objects (without concern for whether they are already attached). Progressive verbs can function as nouns, verbs and adjectives and once again a naïve approximation to the correct attachment is made. Any progressive verb which appears after a determiner or quantifier is considered a noun. Otherwise, it is a verb and passes 3 and 4 are repeated to attach subjects and objects.

Finally, SEXTANT collapses the nn, nnprep and adj relations together into a single broad noun-modifier grammatical relation. Grefenstette (1994) claims this extractor has a grammatical relation accuracy of 75% after manually checking 60 sentences.

7 Approach

Our approach uses voting across the known supersenses of automatically extracted synonyms, to select a supersense for the unknown nouns. This technique is similar to Hearst and Schütze (1993) and Widdows (2003). However, sometimes the unknown noun does not appear in our 2 billion word corpus, or at least does not appear frequently enough to provide sufficient contextual information to extract reliable synonyms. In these cases, our fall-back method is a simple hand-coded classifier which examines the unknown noun and makes a guess based on simple morphological analysis of the suffix. These rules were created by inspecting the suffixes of rare nouns in WORDNET 1.6. The supersense guessing rules are given in Table 5. If none of the rules match, then the default supersense artifact is assigned.

SUFFIX	EXAMPLE	SUPERSENSE
-ness	remoteness	attribute
-tion, -ment	annulment	act
-ist, -man	statesman	person
-ing, -ion	bowling	act
-ity	viscosity	attribute
-ics, -ism	electronics	cognition
-ene, -ane, -ine	arsine	substance
-er, -or, -ic, -ee, -an	mariner	person
-gy	entomology	cognition

Table 5: Hand-coded rules for supersense guessing

The problem now becomes how to convert the ranked list of extracted synonyms for each unknown noun into a single supersense selection. Each extracted synonym votes for its one or more supersenses that appear in WORDNET 1.6. There are many parameters to consider:

- how many extracted synonyms to use;
- how to weight each synonym's vote;
- whether unreliable synonyms should be filtered out;
- how to deal with polysemous synonyms.

The experiments described below consider a range of options for these parameters. In fact, these experiments are so quick to run we have been able to exhaustively test many combinations of these parameters. We have experimented with up to 200 voting extracted synonyms.

There are several ways to weight each synonym's contribution. The simplest approach would be to give each synonym the same weight. Another approach is to use the scores returned by the similarity system. Alternatively, the weights can use the ranking of the extracted synonyms. Again these options have been considered below. A related question is whether to use all of the extracted synonyms, or perhaps filter out synonyms for which a small amount of contextual information has been extracted, and so might be unreliable.

The final issue is how to deal with polysemy. Does every supersense of each extracted synonym get the whole weight of that synonym or is it distributed evenly between the supersenses like Resnik (1995)? Another alternative is to only consider unambiguous synonyms with a single supersense in WORDNET.

A disadvantage of this similarity approach is that it requires full synonym extraction, which compares the unknown word against a large number of words when, in

SYSTEM	WN 1.6	WN 1.7.1
Ciaramita and Johnson baseline	21%	28%
Ciaramita and Johnson perceptron	53%	53%
Similarity based results	**68**%	**63**%

Table 6: Summary of supersense tagging accuracies

fact, we want to calculate the similarity to a small number of supersenses. This inefficiency could be reduced significantly if we consider only very high frequency words, but even this is still expensive.

8 Results

We have used the WORDNET 1.6 test set to experiment with different parameter settings and have kept the WORDNET 1.7.1 test set as a final comparison of best results with Ciaramita and Johnson (2003). The experiments were performed by considering all possible configurations of the parameters described above.

The following voting options were considered for each supersense of each extracted synonym: the initial voting weight for a supersense could either be a constant (IDENTITY) or the similarity score (SCORE) of the synonym. The initial weight could then be divided by the number of supersenses to share out the weight (SHARED). The weight could also be divided by the rank (RANK) to penalise supersenses further down the list. The best performance on the 1.6 test set was achieved with the SCORE voting, without sharing or ranking penalties.

The extracted synonyms are filtered before contributing to the vote with their supersense(s). This filtering involves checking that the synonym's frequency and number of contexts are large enough to ensure it is reliable. We have experimented with a wide range of cutoffs and the best performance on the 1.6 test set was achieved using a minimum cutoff of 5 for the synonym's frequency and the number of contexts it appears in.

The next question is how many synonyms are considered. We considered using just the nearest unambiguous synonym, and the top 5, 10, 20, 50, 100 and 200 synonyms. All of the top performing configurations used 50 synonyms. We have also experimented with filtering out highly polysemous nouns by eliminating words with two, three or more synonyms. However, such a filter turned out to make little difference.

Finally, we need to decide when to use the similarity measure and when to fall-back to the guessing rules. This is determined by looking at the frequency and number of attributes for the unknown word. Not surprisingly, the similarity system works better than the guessing rules if it has any information at all.

The results are summarised in Table 6. The accuracy of the best-performing configurations was 68% on the WORDNET 1.6 test set with several other parameter combinations described above performing nearly as well. On the previously unused WORDNET 1.7.1 test set, our accuracy is 63% using the best system on the WORDNET 1.6 test set. By optimising the parameters on the 1.7.1 test set we can increase that to 64%, indicating that we have not excessively over-tuned on the 1.6 test set. Our results significantly outperform Ciaramita and Johnson (2003) on both test sets even though our system is unsupervised. The large difference between our 1.6 and 1.7.1 test set accuracy demonstrates that the 1.7.1 set is much harder.

	WORDNET 1.6				WORDNET 1.7.1			
SUPERSENSE	N	P	R	F	N	P	R	F
Tops	2	0	0	0	1	50	100	67
act	84	60	74	66	86	53	73	61
animal	16	69	56	62	5	33	60	43
artifact	134	61	86	72	129	57	76	65
attribute	32	52	81	63	16	44	69	54
body	8	88	88	88	5	50	40	44
cognition	31	56	45	50	41	70	34	46
communication	66	80	56	66	57	58	44	50
event	14	83	36	50	10	80	40	53
feeling	8	70	88	78	1	0	0	0
food	29	91	69	78	12	67	67	67
group	27	75	22	34	26	50	4	7
location	43	81	30	44	13	40	15	22
motive	0	0	0	0	1	0	0	0
object	17	73	47	57	13	75	23	35
person	155	76	89	82	207	81	86	84
phenomenon	3	100	100	100	9	0	0	0
plant	11	80	73	76	0	0	0	0
possession	9	100	22	36	16	78	44	56
process	2	0	0	0	9	50	11	18
quantity	12	80	33	47	5	0	0	0
relation	2	100	50	67	0	0	0	0
shape	1	0	0	0	0	0	0	0
state	21	48	48	48	28	50	39	44
substance	24	58	58	58	44	63	73	67
time	5	100	60	75	10	36	40	38
Overall	756	68	68	68	744	63	63	63

Table 7: Breakdown of results by supersense

Table 7 shows the breakdown in performance for each supersense. The columns show the number of instances of each supersense with the precision, recall and f-score measures as percentages. The most frequent supersenses in both test sets were person, attribute and act. Of the frequent categories, person is the easiest supersense to get correct in both the 1.6 and 1.7.1 test sets, followed by food, artifact and substance. This is not surprising since these concrete words tend to have very fewer other senses, well constrained contexts and a relatively high frequency. These factors are conducive for extracting reliable synonyms.

These results also support Ciaramita and Johnson's view that abstract concepts like communication, cognition and state are much harder. We would expect the location supersense to perform well since it is quite concrete, but unfortunately our synonym extraction system does not incorporate proper nouns, so many of these words were classified using the hand-built classifier. Also, in the data from Ciaramita and Johnson all of the words are in lower case, so no sensible guessing rules could help.

9 Other Alternatives and Future Work

An alternative approach worth exploring is to create context vectors for the supersense categories themselves and compare these against the words. This has the advantage of producing a much smaller number of vectors to compare against. In the current system, we must compare a word against the entire vocabulary (over 500 000 headwords), which is much less efficient than a comparison against only 26 supersense context vectors.

The question now becomes how to construct vectors of supersenses. The most obvious solution is to sum the context vectors across the words which have each supersense. However, our early experiments suggest that this produces extremely large vectors which do not match well against the much smaller vectors of each unseen word. Also, the same questions arise in the construction of these vectors. How are words with multiple supersenses handled? Our preliminary experiments suggest that only combining the vectors for unambiguous words produces the best results.

One solution would be to take the intersection between vectors across words for each supersense (i.e. to find the common contexts that these words appear in). However, given the sparseness of the data this may not leave very large context vectors. A final solution would be to consider a large set of the *canonical attributes* (Curran and Moens, 2002a) to represent each supersense. Canonical attributes summarise the key contexts for each headword and are used to improve the efficiency of the similarity comparisons.

There are a number of problems our system does not currently handle. Firstly, we do not include proper names in our similarity system which means that location entities can be very difficult to identify correctly (as the results demonstrate). Further, our similarity system does not currently incorporate multi-word terms. We overcome this by using the synonyms of the last word in the multi-word term. However, there are 174 multi-word terms (23%) in the WORDNET 1.7.1 test set which we could probably tag more accurately with synonyms for the whole multi-word term. Finally, we plan to implement a supervised machine learner to replace the fallback method, which currently has an accuracy of 37% on the WORDNET 1.7.1 test set.

We intend to extend our experiments beyond the Ciaramita and Johnson (2003) set to include previous and more recent versions of WORDNET to compare their difficulty, and also perform experiments over a range of corpus sizes to determine the impact of corpus size on the quality of results.

We would like to move onto the more difficult task of insertion into the hierarchy itself and compare against the initial work by Widdows (2003) using latent semantic analysis. Here the issue of how to combine vectors is even more interesting since there is the additional structure of the WORDNET inheritance hierarchy and the small synonym sets that can be used for more fine-grained combination of vectors.

10 Conclusion

Our application of semantic similarity to supersense tagging follows earlier work by Hearst and Schütze (1993) and Widdows (2003). To classify a previously unseen common noun our approach extracts synonyms which vote using their supersenses in WORDNET 1.6. We have experimented with several parameters finding that the best configuration uses 50 extracted synonyms, filtered by frequency and number of contexts to increase their reliability. Each synonym votes for each of its supersenses from WORDNET 1.6 using the similarity score from our synonym extractor.

Using this approach we have significantly outperformed the supervised multi-class perceptron Ciaramita and Johnson (2003). This paper also demonstrates the use of a very efficient shallow NLP pipeline to process a massive corpus. Such a corpus is needed to acquire reliable contextual information for the often very rare nouns we are attempting to supersense tag. This application of semantic similarity demonstrates that an unsupervised methods can outperform supervised methods for some NLP tasks if enough data is available.

Acknowledgements

We would like to thank Massi Ciaramita for supplying his original data for these experiments and answering our queries, and to Stephen Clark and the anonymous reviewers for their helpful feedback and corrections. This work has been supported by a Commonwealth scholarship, Sydney University Travelling Scholarship and Australian Research Council Discovery Project DP0453131.

References

L. Douglas Baker and Andrew McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 96–103, Melbourne, Australia.

Doug Beeferman. 1998. Lexical discovery with an enriched semantic network. In *Proceedings of the Workshop on Usage* *of WordNet in Natural Language Processing Systems*, pages 358–364, Montréal, Québec, Canada.

Thorsten Brants. 2000. TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 224–231, Seattle, WA USA.

Anita Burgun and Olivier Bodenreider. 2001. Comparing terms, concepts and semantic classes in WordNet and the Unified Medical Language System. In *Proceedings of the Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pages 77–82, Pittsburgh, PA USA.

Sharon A. Caraballo and Eugene Charniak. 1999. Determining the specificity of nouns from text. In *Proceedings of the Joint ACL SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 63–70, College Park, MD USA.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense tagging of unknown nouns in WordNet. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 168–175, Sapporo, Japan.

Massimiliano Ciaramita, Thomas Hofmann, and Mark Johnson. 2003. Hierarchical semantic classification: Word sense disambiguation with world knowledge. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico.

Massimiliano Ciaramita. 2002. Boosting automatic lexical acquisition with morphological information. In *Proceedings of the Workshop on Unsupervised Lexical Acquisition*, pages 17–25, Philadelphia, PA, USA.

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*, 28(2):187–206, June.

Koby Crammer and Yoram Singer. 2001. Ultraconservative online algorithms for multiclass problems. In *Proceedings of the 14th annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory*, pages 99–115, Amsterdam, The Netherlands.

James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 91–98, Budapest, Hungary.

James R. Curran and Marc Moens. 2002a. Improvements in automatic thesaurus extraction. In *Proceedings of the Workshop on Unsupervised Lexical Acquisition*, pages 59–66, Philadelphia, PA, USA.

James R. Curran and Marc Moens. 2002b. Scaling context space. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 231–238, Philadelphia, PA, USA.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA USA.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Boston, MA USA.

Marti A. Hearst and Hinrich Schütze. 1993. Customizing a lexicon to better suit a computational task. In *Proceedings of the Workshop on Acquisition of Lexical Knowledge from Text*, pages 55–69, Columbus, OH USA.

Rob Koeling. 2000. Chunking with maximum entropy models. In *Proceedings of the 4th Conference on Computational Natural Language Learning and of the 2nd Learning Language in Logic Workshop*, pages 139–141, Lisbon, Portugal.

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.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

Tom Morton. 2002. Grok tokenizer. *Grok OpenNLP toolkit*.

Marius Pasca and Sanda M. Harabagiu. 2001. The informative role of WordNet in open-domain question answering. In *Proceedings of the Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pages 138–143, Pittsburgh, PA USA.

Darren Pearce. 2001. Synonymy in collocation extraction. In *Proceedings of the Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pages 41–46, Pittsburgh, PA USA.

Philip Resnik. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, Montreal, Canada.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington, D.C. USA.

Hinrich Schütze. 1992. Context space. In *Intelligent Probabilistic Approaches to Natural Language*, number FS-92-04 in Fall Symposium Series, pages 113–120, Stanford University, CA USA.

Dominic Widdows. 2003. Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 276–283, Edmonton, Alberta Canada.

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*, pages 454–460, Nantes, France.

Learning Semantic Classes for Word Sense Disambiguation

Upali S. Kohomban **Wee Sun Lee**
Department of Computer Science
National University of Singapore
Singapore, 117584
{upalisat,leews}@comp.nus.edu.sg

Abstract

Word Sense Disambiguation suffers from a long-standing problem of knowledge acquisition bottleneck. Although state of the art supervised systems report good accuracies for selected words, they have not been shown to be promising in terms of scalability. In this paper, we present an approach for learning coarser and more general set of concepts from a sense tagged corpus, in order to alleviate the knowledge acquisition bottleneck. We show that these general concepts can be transformed to fine grained word senses using simple heuristics, and applying the technique for recent SENSEVAL data sets shows that our approach can yield state of the art performance.

1 Introduction

Word Sense Disambiguation (WSD) is the task of determining the meaning of a word in a given context. This task has a long history in natural language processing, and is considered to be an intermediate task, success of which is considered to be important for other tasks such as Machine Translation, Language Understanding, and Information Retrieval.

Despite a long history of attempts to solve WSD problem by empirical means, there is not any clear consensus on what it takes to build a high performance implementation of WSD. Algorithms based on Supervised Learning, in general, show better performance compared to unsupervised systems. But they suffer from a serious drawback: the difficulty of acquiring considerable amounts of training data, also known as *knowledge acquisition bottleneck*. In the typical setting, supervised learning needs training data created for each and every polysemous word; Ng (1997) estimates an effort of 16 person-years for acquiring training data for 3,200 significant words in English. Mihalcea and Chklovski (2003) provide a similar estimate of an 80 person-year effort for creating manually labelled training data for about 20,000 words in a common English dictionary.

Two basic approaches have been tried as solutions to the lack of training data, namely unsupervised systems and semi-supervised bootstrapping techniques. Unsupervised systems mostly work on knowledge-based techniques, exploiting sense knowledge encoded in machine-readable dictionary entries, taxonomical hierarchies such as WORDNET (Fellbaum, 1998), and so on. Most of the bootstrapping techniques start from a few 'seed' labelled examples, classify some unlabelled instances using this knowledge, and iteratively expand their knowledge using information available within newly labelled data. Some others employ hierarchical relatives such as hypernyms and hyponyms.

In this work, we present another practical alternative: we reduce the WSD problem to a one of finding generic semantic class of a given word instance. We show that learning such classes can help relieve the problem of knowledge acquisition bottleneck.

1.1 Learning senses as concepts

As the semantic classes we propose learning, we use WORDNET lexicographer file identifiers corresponding to each fine-grained sense. By learning these generic classes, we show that we can reuse training data, without having to rely on specific training data for each word. This can be done because the semantic classes are common to words unlike senses; for learning the properties of a given class, we can use the data from various words. For instance, the noun *crane* falls into two semantic classes ANIMAL and ARTEFACT. We can expect the words such as *pelican* and *eagle* (in the bird sense) to have similar usage patterns to those of ANIMAL sense of *crane*, and to provide common training examples for that particular class.

For learning these classes, we can make use of any training example labelled with WORDNET senses for supervised WSD, as we describe in section 3.1.

Once the classification is done for an instance, the resulting semantic classes can be transformed into finer grained senses using some heuristical mapping, as we show in the next sub section. This would not guarantee a perfect conversion because such a mapping can miss some finer senses, but as we show in what follows, this problem in itself does not prevent us from attaining good performance in a practical WSD setting.

1.2 Information loss in coarse grained senses

As an empirical verification of the hypothesis that we can still build effective fine-grained sense disambiguators despite the loss of information, we analyzed the performance of a hypothetical coarse grained classifier that can perform at 100% accuracy. As the general set of classes, we used WORDNET unique beginners, of which there are 25 for nouns, and 15 for verbs.

To simulate this classifier on SENSEVAL English all-words tasks' data (Edmonds and Cotton, 2001; Snyder and Palmer, 2004), we mapped the fine-grained senses from official answer keys to their respective beginners. There is an information loss in this mapping, because each unique beginner can typically include more than one sense. To see how this 'classifier' fares in a fine-grained task, we can map the 'answers' back to WORDNET fine-grained senses by picking up the sense with the lowest sense number that falls within each unique beginner. In principal, this is the most likely sense within the class, because WORDNET senses are said to be ordered in descending order of frequency. Since this sense is not necessarily the same as the original sense of the instance, the accuracy of the fine-grained answers will be below 100%.

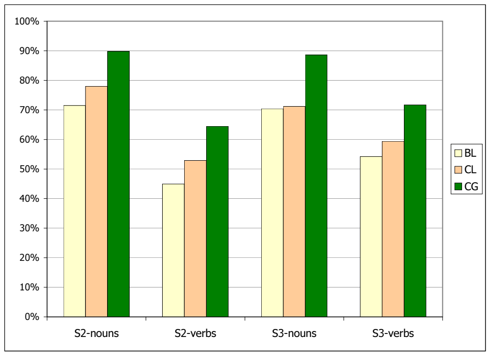


Figure 1: Performance of a hypothetical coarse-grained classifier, output mapped to fine-grained senses, on SENSEVAL English all-words tasks.

Figure 1 shows the performance of this transformed fine-grained classifier (CG) for nouns and verbs with SENSEVAL-2 and 3 English all words task data (marked as S2 and S3 respectively), along with the baseline WORDNET first sense (BL), and the best-performer classifiers at each SENSEVAL excercise (CL), SMUaw (Mihalcea, 2002) and GAMBL-AW (Decadt et al., 2004) respectively.

There is a considerable difference in terms of improvement over baseline, between the state-of-the-art systems and the hypothetical optimal coarse-grained system. This shows us that there is an improvement in performance that we can attain over the state-of-the-art, if we can create a classifier for even a very coarse level of senses, with sufficiently high accuracy. We believe that the chances for such a high accuracy in a coarse-grained sense classifier is better, for several reasons:

- previously reported good performance for coarse grained systems (Yarowsky, 1992)
- better availability of data, due to the possibility of reusing data created for different words. For instance, labelled data for the noun '*crane*' is not found in SEMCOR corpus at all, but there are more than 1000 sample instances for the concept ANIMAL, and more than 9000 for ARTEFACT.

- higher inter-annotator agreement levels and lower corpus/genre dependencies in training/testing data due to coarser senses.

1.3 Overall approach

Basically, we assume that we can learn the 'concepts', in terms of WORDNET unique beginners, using a set of data labelled with these concepts, regardless of the actual word that is labelled. Hence, we can use a generic data set that is large enough, where various words provide training examples for these concepts, instead of relying upon data from the examples of the same word that is being classified.

Unfortunately, simply labelling each instance with its semantic class and then using standard supervised learning algorithms did not work well. This is probably because the effectiveness of the feature patterns often depend on the actual word being disambiguated and not just its semantic class. For example, the phrase *'run the newspaper'* effectively indicates that *'newspaper'* belongs to the semantic class GROUP. But *'run the tape'* indicates that *'tape'* belongs to the semantic class ARTEFACT. The collocation *'run the'* is effective for indicating the GROUP sense only for *'newspaper'* and closely related words such as *'department'* or *'school'*.

In this experiment, we use a k-nearest neighbor classifier. In order to allow training examples of different words from the same semantic class to effectively provide information for each other, we modify the distance between instances in a way that makes the distance between instances of similar words smaller. This is described in Section 3.

The rest of the paper is organized as follows: In section 2, we discuss several related work. We proceed on to a detailed description of our system in section 3, and discuss the empirical results in section 4, showing that our representation can yield state of the art performance.

2 Related Work

Using generic classes as word senses has been done several times in WSD, in various contexts. Resnik (1997) described a method to acquire a set of conceptual classes for word senses, employing *selectional preferences*, based on the idea that certain linguistic predicates constraint the semantic interpretation of underlying words into certain classes. The method he proposed could acquire these constraints from a raw corpus automatically.

Classification proposed by Levin (1993) for English verbs remains a matter of interest. Although these classes are based on syntactic properties unlike those in WORDNET, it has been shown that they can be used in automatic classifications (Stevenson and Merlo, 2000). Korhonen (2002) proposed a method for mapping WORDNET entries into Levin classes.

WSD System presented by Crestan et al. (2001) in SENSEVAL-2 classified words into WORDNET unique beginners. However, their approach did not use the fact that the primes are common for words, and training data can hence be reused.

Yarowsky (1992) used Roget's Thesaurus categories as classes for word senses. These classes differ from those mentioned above, by the fact that they are based on topical context rather than syntax or grammar.

3 Basic Design of the System

The system consists of three classifiers, built using local context, part of speech and syntax-based relationships respectively, and combined with the most-frequent sense classifier by using weighted majority voting. Our experiments (section 4.3) show that building separate classifiers from different subsets of features and combining them works better than building one classifier by concatenating the features together.

For training and testing, we used publicly available data sets, namely SEMCOR corpus (Miller et al., 1993) and SENSEVAL English all-words task data. In order to evaluate the systems performance *in vivo*, we mapped the outputs of our classifier to the answers given in the key. Although we face a penalty here due to the loss of granularity, this approach allows a direct comparison of actual usability of our system.

3.1 Data

As training corpus, we used Brown-1 and Brown-2 parts of SEMCOR corpus; these parts have all of their open-class words tagged with corresponding WORDNET senses. A part of the training corpus was set aside as the development corpus. This part was selected by randomly selecting a portion of multiclass words (600 instances for each part of speech) from the training data set. As labels, the semantic class (lexicographic file number) was extracted from the sense key of each instance. Testing data sets from SENSEVAL-2 and SENSEVAL-3 English all-words tasks were used as testing corpora.

3.2 Features

The feature set we selected was fairly simple; As we understood from our initial experiments, wide-window context features and topical context were not of much use for learning semantic classes from a multi-word training data set. Instead of generalizing, wider context windows add to noise, as seen from validation experiments with held-out data.

Following are the features we used:

3.2.1 Local context

This is a window of n words to the left, and n words to the right, where $n \in \{1, 2, 3\}$ is a parameter we selected via cross validation.[1]

Punctuation marks were removed and all words were converted into lower case. The feature vector was calculated the same way for both nouns and verbs. The window did not exceed the boundaries of a sentence; when there were not enough words to either side of the word within the window, the value `NULL` was used to fill the remaining positions.

For instance, for the noun *'companion'* in sentence (given with POS tags)

> *'Henry/NNP peered/VBD doubtfully/RB at/IN his/PRP$ drinking/NN companion/NN through/IN bleary/JJ ,/, tear-filled/JJ eyes/NNS ./.'*

the local context feature vector is `[at, his, drinking, through, bleary, tear-filled]`, for window size $n = 3$. Notice that we did not consider the hyphenated words as two words, when the data files had them annotated as a single token.

3.2.2 Part of speech

This consists of parts of speech for a window of n words to both sides of word (excluding the word itself), with quotation signs and punctuation marks ignored. For SEMCOR files, existing parts of speech were used; for SENSEVAL data files, parts of speech from the accompanying Penn-Treebank parsed data files were aligned with the XML data files. The value vector is calculated the same way as the local context, with the same constraint on sentence boundaries, replacing vacancies with `NULL`.

As an example, for the sentence we used in the previous example, the part-of-speech vector with context size $n = 3$ for the verb *peered* is `[NULL, NULL, NNP, RB, IN, PRP$]`.

[1]Validation results showed that a window of two words to both sides yields the best performance for both local context and POS features. $n = 2$ is the size we used in actual evaluation.

Feature	Example	Value
nouns		
Subject - verb	[art] represents a culture	`represent`
Verb - object	He sells his [art]	`sell`
Adjectival modifiers	the ancient [art] of runes	`ancient`
Prepositional connectors	academy of folk [art]	`academy of`
Post-nominal modifiers	the [art] of fishing	`of fishing`
verbs		
Subject - verb	He [sells] his art	`he`
Verb - object	He [sells] his art	`art`
Infinitive connector	He will [sell] his art	`he`
Adverbial modifier	He can [paint] well	`well`
Words in split infinitives	to boldly [go]	`boldly`

Table 1: Syntactic relations used as features. The target word is shown inside [brackets]

3.2.3 Syntactic relations with the word

The words that hold several kinds of syntactic relations with the word under consideration were selected. We used Link Grammar parser due to Sleator and Temperley (1991) because of the information-rich parse results produced by it.

Sentences in SEMCOR corpus files and the SENSEVAL files were parsed with Link parser, and words were aligned with links. A given instance of a word can have more than one syntactic features present. Each of these features was considered as a binary feature, and a vector of binary values was constructed, of which each element denoted a unique feature found in the test set of the word.

Each syntactic pattern feature falls into either of two types *collocation* or *relation*:

Collocation features Collocation features are such features that connect the word under consideration to another word, with a preposition or an infinitive in between — for instance, the phrase *'art of change-ringing'* for the word *art*. For these features, the feature value consists of two words, which are connected to the given word either from left or from right, in a given order. For the above example, the feature value is `[∼.of.change-ringing]`, where $\sim$ denotes the placeholder for word under consideration.

Relational features Relational features represent more direct grammatical relationships, such as subject-verb or noun-adjective, the word under consideration has with surrounding words. When encoding the feature value, we specified the relation type and the value of the feature in the given instance. For instance, in the phrase *'Henry peered doubtfully'*, the adverbial modifier feature for the verb 'peered' is encoded as `[adverb-mod doubtfully]`.

A description of the relations for each part of speech is given in the table 1.

3.3 Classifier and instance weighting

The classifier we used was TiMBL, a memory based learner due to Daelemans et al. (2003). One reason for this choice was that memory based learning has shown to perform well in previous word sense disambiguation tasks, including some best performers in SENSEVAL, such as (Hoste et al., 2001; Decadt et al., 2004; Mihalcea and Faruque, 2004). Another reason is that TiMBL supported exemplar weights, a necessary feature for our system for the reasons we describe in the next section.

One of the salient features of our system is that it does not consider every example to be equally important. Due to the fact that training instances from different instances can provide confusing examples, as shown in section 1.3, such an approach cannot be trusted to give good performance; we verified this by our own findings through empirical evaluations as shown in section 4.2.

3.3.1 Weighting instances with similarity

We use a similarity based measure to assign weights to training examples. In the method we use, these weights are used to adjust the distances between the test instance and the example instances. The distances are adjusted according to the formula

$$\Delta^E(X,Y) = \frac{\Delta(X,Y)}{ew_X + \epsilon} \quad ,$$

where $\Delta^E(X,Y)$ is the adjusted distance between instance Y and example X, $\Delta(X,Y)$ is the original distance, ew_X is the exemplar weight of instance X. The small constant ϵ is added to avoid division by zero.

There are various schemes used to measure inter-sense similarity. Our experiments showed that the measure defined by Jiang and Conrath (1997) (JCn) yields best results. Results for various weighting schemes are discussed in section 4.2.

3.3.2 Instance weighting explained

The exemplar weights were derived from the following method:

1. pick a labelled example e, and extract its sense s_e and semantic class c_e.

2. if the class c_e is a candidate for the current test word w, i.e. w has any senses that fall into c_e, find out the most frequent sense of w, s_w^{ce}, within c_e. We define the most frequent sense within a class as the sense that has the lowest WORDNET sense number within that class. If none of the senses of w fall into c_e, we ignore that example.

3. calculate the relatedness measure between s_e and s_w^{ce}, using whatever the similarity metric being considered. This is the exemplar weight for example e.

In the implementation, we used freely available `WordNet::Similarity` package (Pedersen et al., 2004). [2]

3.4 Classifier optimization

A part of SEMCOR corpus was used as a validation set (see section 3.1). The rest was used as training data in validation phase. In the preliminary experiments, it was seen that the generally recommended classifier options yield good enough performance, although variations of switches could improve performance slightly in certain cases. Classifier options were selected by a search over the available option space for only three basic classifier parameters, namely, number of nearest neighbors, distance metric and feature weighting scheme.

[2]`WordNet::Similarity` is a perl package available freely under GNU General Public Licence. http://wn-similarity.sourceforge.net.

Classifier	Senseval-2	Senseval-3
Baseline	0.617	0.627
POS	0.616	0.614
Local context	0.627	0.633
Synt. Pat	0.620	0.612
Concatenated	0.609	0.611
Combined	**0.631**	**0.643**

Table 2: Results of baseline, individual, and combined classifiers: recall measures for nouns and verbs combined.

4 Results

In what follows, we present the results of our experiments in various test cases.[3] We combined the three classifiers and the WORDNET first-sense classifier through simple majority voting. For evaluating the systems with SENSEVAL data sets, we mapped the outputs of our classifiers to WORDNET senses by picking the most-frequent sense (the one with the lowest sense number) within each of the class. This mapping was used in all tests. For all evaluations, we used SENSEVAL official scorer.

We could use the setting only for nouns and verbs, because the similarity measures we used were not defined for adjectives or adverbs, due to the fact that hypernyms are not defined for these two parts of speech. So we list the initial results only for nouns and verbs.

4.1 Individual classifiers vs. combination

We evaluated the results of the individual classifiers before combination. Only local context classifier could outperform the baseline in general, although there is a slight improvement with the syntactic pattern classifier on SENSEVAL-2 data.

The results are given in the table 2, together with the results of voted combination, and baseline WORDNET first sense. Classifier shown as 'concatenated' is a single classifier trained from all of these feature vectors concatenated to make a single vector. Concatenating features this way does not seem to improve performance. Although exact reasons for this are not clear, this is consistent with previous observations (Hoste et al., 2001; Decadt et al., 2004) that combining classifiers, each using different features, can yield good performance.

[3]Note that the experiments and results are reported for SENSEVAL data for comparison purposes, and were not involved in parameter optimization, which was done with the development sample.

	Senseval-2	Senseval-3
No similarity used	0.608	0.599
Resnik	0.540	0.522
JCn	**0.631**	**0.643**

Table 3: Effect of different similarity schemes on recall, combined results for nouns and verbs

	Senseval-2	Senseval-3
SM	0.631	0.643
GW	0.634	0.649
LW	**0.641**	**0.650**

Table 4: Improvement of performance with classifier weighting. Combined results for nouns and verbs with voting schemes Simple Majority (SM), Global classifier weights (GW) and local weights (LW).

4.2 Effect of similarity measure

Table 3 shows the effect of JCn and Resnik similarity measures, along with no similarity weighting, for the combined classifier. It is clear that proper similarity measure has a major impact on the performance, with Resnik measure performing worse than the baseline.

4.3 Optimizing the voting process

Several voting schemes were tried for combining classifiers. Simple majority voting improves performance over baseline. However, previously reported results such as (Hoste et al., 2001) and (Decadt et al., 2004) have shown that optimizing the voting process helps improve the results. We used a variation of Weighted Majority Algorithm (Littlestone and Warmuth, 1994). The original algorithm was formulated for binary classification tasks; however, our use of it for multi-class case proved to be successful.

We used the held-out development data set for adjusting classifier weights. Originally, all classifiers have the same weight of 1. With each test instance, the classifier builds the final output considering the weights. If this output turns out to be wrong, the classifiers that contributed to the wrong answer get their weights reduced by some factor. We could ad-

	Senseval-2	Senseval-3
System	0.777	0.806
Baseline	0.756	0.783

Table 5: Coarse grained results

just the weights locally or globally; In global setting, the weights were adjusted using a random sample of held-out data, which contained different words. These weights were used for classifying all words in the actual test set. In local setting, each classifier weight setting was optimized for individual words that were present in test sets, by picking up random samples of the same word from SEMCOR .[4] Table 4 shows the improvements with each setting.

Coarse grained (at semantic-class level) results for the same system are shown in table 5. Baseline figures reported are for the most-frequent class.

4.4 Final results on SENSEVAL data

Here, we list the performance of the system with adjectives and adverbs added for the ease of comparison. Due to the facts mentioned at the beginning of this section, our system was not applicable for these parts of speech, and we classified all instances of these two POS types with their most frequent sense. We also identified the multi-word phrases from the test documents. These phrases generally have a unique sense in WORDNET ; we marked all of them with their first sense without classifying them. All the multiple-class instances of nouns and verbs were classified and converted to WORDNET senses by the method described above, with locally optimized classifier voting.

The results of the systems are shown in tables 7 and 8. Our system's results in both cases are listed as Simil-Prime, along with the baseline WORDNET first sense (including multi-word phrases and 'U' answers), and the two best performers' results reported.[5] These results compare favorably with the official results reported in both tasks.

[4]Words for which there were no samples in SEMCOR were classified using a weight of 1 for all classifiers.

[5]The differences of the baseline figures from the previously reported figures are clearly due to different handling of multiword phrases, hyphenated words, and unknown words in each system. We observed by analyzing the answer keys that even better baseline figures are technically possible, with better techniques to identify these special cases.

	Senseval-2	Senseval-3
Micro Average	< 0.0001	< 0.0001
Macro Average	0.0073	0.0252

Table 6: One tailed paired t-test significance levels of results: $P(T \leqslant t)$

System	Recall
SMUaw (Mihalcea, 2002)	0.690
Simil-Prime	0.664
Baseline (WORDNET first sense)	0.648
CNTS-Antwerp (Hoste et al., 2001)	0.636

Table 7: Results for SENSEVAL-2 English all words data for all parts of speech and fine grained scoring.

Significance of results To verify the significance of these results, we used one-tailed paired t-test, using results of baseline WORDNET first sense and our system as pairs. Tests were done both at micro-average level and macro-average level, (considering test data set as a whole and considering per-word average). Null hypothesis was that there is no significant improvement over the baseline. Both settings yield good significance levels, as shown in table 6.

5 Conclusion and Future Work

We analyzed the problem of *Knowledge Acquisition Bottleneck* in WSD, proposed using a general set of semantic classes as a trade-off, and discussed why such a system is promising. Our formulation allowed us to use training examples from words different from the actual word being classified. This makes the available labelled data reusable for different words, relieving the above problem. In order to facilitate learning, we introduced a technique based on word sense similarity.

The generic classes we learned can be mapped to

System	Recall
Simil-Prime	0.661
GAMBL-AW-S (Decadt et al., 2004)	0.652
SenseLearner (Mihalcea and Faruque, 2004)	0.646
Baseline (WORDNET first sense)	0.642

Table 8: Results for SENSEVAL-3 English all words data for all parts of speech and fine grained scoring.

finer grained senses with simple heuristics. Through empirical findings, we showed that our system can attain state of the art performance, when applied to standard fine-grained WSD evaluation tasks.

In the future, we hope to improve on these results: Instead of using WORDNET unique beginners, using more natural semantic classes based on word usage would possibly improve the accuracy, and finding such classes would be a worthwhile area of research. As seen from our results, selecting correct similarity measure has an impact on the final outcome. We hope to work on similarity measures that are more applicable in our task.

6 Acknowledgements

Authors wish to thank the three anonymous reviewers for their helpful suggestions and comments.

References

E. Crestan, M. El-Bèze, and C. De Loupy. 2001. Improving wsd with multi-level view of context monitored by similarity measure. In *Proceeding of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems*, Toulouse, France.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2003. TiMBL: Tilburg Memory Based Learner, version 5.0, reference guide. Technical report, ILK 03-10.

Bart Decadt, Véronique Hoste, Walter Daelemans, and Antal Van den Bosch. 2004. GAMBL, genetic algorithm optimization of memory-based wsd. In *Senseval-3: Third Intl. Workshop on the Evaluation of Systems for the Semantic Analysis of Text*.

P. Edmonds and S. Cotton. 2001. Senseval-2: Overview. In *Proc. of the Second Intl. Workshop on Evaluating Word Sense Disambiguation Systems (Senseval-2)*.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA.

Véronique Hoste, Anne Kool, and Walter Daelmans. 2001. Classifier optimization and combination in English all words task. In *Proceeding of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems*.

J. Jiang and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research in Computational Linguistics*.

Anna Korhonen. 2002. Assigning verbs to semantic classes via wordnet. In *Proceedings of the COLING Workshop on Building and Using Semantic Networks*.

Beth Levin. 1993. *English Verb Classes and Alternations*. University of Chicago Press, Chicago, IL.

N Littlestone and M.K. Warmuth. 1994. The weighted majority algorithm. *Information and Computation*, 108(2):212–261.

Rada Mihalcea and Tim Chklovski. 2003. Open Mind Word Expert: Creating large annotated data collections with web users' help. In *Proceedings of the EACL 2003 Workshop on Linguistically Annotated Corpora*.

Rada Mihalcea and Ehsanul Faruque. 2004. Senselearner: Minimally supervised word sense disambiguation for all words in open text. In *Senseval-3: Third Intl. Workshop on the Evaluation of Systems for the Semantic Analysis of Text*.

Rada Mihalcea. 2002. Bootstrapping large sense tagged corpora. In *Proc. of the 3rd Intl. Conference on Languages Resources and Evaluations*.

G. Miller, C. Leacock, T. Randee, and R. Bunker. 1993. A semantic concordance. In *Proc. of the 3rd DARPA Workshop on Human Language Technology*.

Hwee Tou Ng. 1997. Getting serious about word sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, pages 1–7.

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. Wordnet::Similarity - Measuring the relatedness of concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*.

P. Resnik. 1997. Selectional preference and sense disambiguation. In *Proc. of ACL Siglex Workshop on Tagging Text with Lexical Semantics, Why, What and How?*

D. Sleator and D. Temperley. 1991. Parsing English with a Link Grammar. Technical report, Carnegie Mellon University Computer Science CMU-CS-91-196.

B. Snyder and M. Palmer. 2004. The English all-words task. In *Senseval-3: Third Intl. Workshop on the Evaluation of Systems for the Semantic Analysis of Text*.

Suzanne Stevenson and Paola Merlo. 2000. Automatic lexical acquisition based on statistical distributions. In *Proc. of the 17th conf. on Computational linguistics*.

David Yarowsky. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of COLING-92*, pages 454–460.

The Role of Semantic Roles in Disambiguating Verb Senses

Hoa Trang Dang
National Institute of Standards and Technology
Gaithersburg, MD 20899
hoa.dang@nist.gov

Martha Palmer
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
mpalmer@cis.upenn.edu

Abstract

We describe an automatic Word Sense Disambiguation (WSD) system that disambiguates verb senses using syntactic and semantic features that encode information about predicate arguments and semantic classes. Our system performs at the best published accuracy on the English verbs of Senseval-2. We also experiment with using the gold-standard predicate-argument labels from PropBank for disambiguating fine-grained WordNet senses and course-grained PropBank framesets, and show that disambiguation of verb senses can be further improved with better extraction of semantic roles.

1 Introduction

A word can have different meanings depending on the context in which it is used. Word Sense Disambiguation (WSD) is the task of determining the correct meaning ("sense") of a word in context, and several efforts have been made to develop automatic WSD systems. Early work on WSD (Yarowsky, 1995) was successful for easily distinguishable homonyms like *bank*, which have multiple unrelated meanings. While homonyms are fairly tractable, highly polysemous verbs, which have related but subtly distinct senses, pose the greatest challenge for WSD systems (Palmer et al., 2001).

Verbs are syntactically complex, and their syntax is thought to be determined by their underlying semantics (Grimshaw, 1990; Levin, 1993). Levin verb classes, for example, are based on the ability of a verb to occur in pairs of syntactic frames (diathesis alternations); different senses of a verb belong to different verb classes, which have different sets of syntactic frames that are supposed to reflect underlying semantic components that constrain allowable arguments. If this is true, then the correct sense of a verb should be revealed (at least partially) in its arguments.

In this paper we show that the performance of automatic WSD systems can be improved by using richer linguistic features that capture information about predicate arguments and their semantic classes. We describe our approach to automatic WSD of verbs using maximum entropy models to combine information from lexical collocations, syntax, and semantic class constraints on verb arguments. The system performs at the best published accuracy on the English verbs of the Senseval-2 (Palmer et al., 2001) exercise on evaluating automatic WSD systems. The Senseval-2 verb instances have been manually tagged with their WordNet sense and come primarily from the Penn Treebank WSJ. The WSJ corpus has also been manually annotated for predicate arguments as part of PropBank (Kingsbury and Palmer, 2002), and the intersection of PropBank and Senseval-2 forms a corpus containing gold-standard annotations of WordNet senses and PropBank semantic role labels. This provides a unique opportunity to investigate the role of predicate arguments in verb sense disambiguation. We show that our system's accuracy improves significantly by adding features from PropBank, which explicitly encodes the predicate-argument information that our original set of syntactic and semantic class features attempted to capture.

2 Basic automatic system

Our WSD system was built to combine information from many different sources, using as much linguistic knowledge as could be gathered automatically by NLP tools. In particular, our goal was to see the extent to which sense-tagging of verbs could be improved by adding features that capture information about predicate-arguments and selectional restrictions.

We used the Mallet toolkit (McCallum, 2002) for learning maximum entropy models with Gaussian priors for all our experiments. In order to extract the linguistic features necessary for the models, all sentences containing the target word were automatically part-of-speech-tagged using a maximum entropy tagger (Ratnaparkhi, 1998) and parsed using the Collins parser (Collins, 1997). In addition, an automatic named entity tagger (Bikel et al., 1997) was run on the sentences to map proper nouns to a small set of semantic classes.[1]

2.1 Topical features

We categorized the possible model features into topical features and several types of local contextual features. Topical features for a verb in a sentence look for the presence of keywords occurring *anywhere* in the sentence and any surrounding sentences provided as context (usually one or two sentences). These features are supposed to show the domain in which the verb is being used, since some verb senses are used in only certain domains. The set of keywords is specific to each verb lemma to be disambiguated and is determined automatically from training data so as to minimize the entropy of the probability of the senses conditioned on the keyword. All alphabetic characters are converted to lower case. Words occuring less than twice in the training data or that are in a stoplist[2] of pronouns, prepositions, and conjunctions are ignored.

[1]The inclusion or omission of a particular company or product implies neither endorsement nor criticism by NIST. Any opinions, findings, and conclusions expressed are the authors' own and do not necessarily reflect those of NIST.

[2]`http://www.d.umn.edu/~tpederse/Group01/WordNet/words.txt`

2.2 Local features

The local features for a verb w in a particular sentence tend to look only within the smallest clause containing w. They include *collocational* features requiring no linguistic preprocessing beyond part-of-speech tagging, *syntactic* features that capture relations between the verb and its complements, and *semantic* features that incorporate information about noun classes for subjects and objects:

Collocational features: Collocational features refer to ordered sequences of part-of-speech tags or word tokens immediately surrounding w. They include:

- unigrams: words w_{-2}, w_{-1}, w_0, w_{+1}, w_{+2} and parts of speech p_{-2}, p_{-1}, p_0, p_{+1}, p_{+2}, where w_i and p_i are at position i relative to w
- bigrams: $w_{-2}w_{-1}$, $w_{-1}w_{+1}$, $w_{+1}w_{+2}$; $p_{-2}p_{-1}$, $p_{-1}p_{+1}$, $p_{+1}p_{+2}$
- trigrams: $w_{-3}w_{-2}w_{-1}$, $w_{-2}w_{-1}w_{+1}$, $w_{-1}w_{+1}w_{+2}$, $w_{+1}w_{+2}w_{+3}$; $p_{-3}p_{-2}p_{-1}$, $p_{-2}p_{-1}p_{+1}$, $p_{-1}p_{+1}p_{+2}$, $p_{+1}p_{+2}p_{+3}$

Syntactic features: The system uses heuristics to extract syntactic elements from the parse for the sentence containing w. Let commander VP be the lowest VP that dominates w and that is not immediately dominated by another VP, and let head VP be the lowest VP dominating w (See Figure 1). Then we define the *subject* of w to be the leftmost NP sibling of commander VP, and a *complement* of w to be a node that is a child of the head VP, excluding NPs whose head is a number or a noun from a list of common temporal nouns ("week", "tomorrow", "Monday", etc.). The system extracts the following binary syntactic features:

- Is the sentence passive?
- Is there a subject, direct object (leftmost NP complement of w), indirect object (second leftmost NP complement of w), or clausal complement (S complement of w)?
- What is the word (if any) that is the particle or head of the subject, direct object, or indirect object?

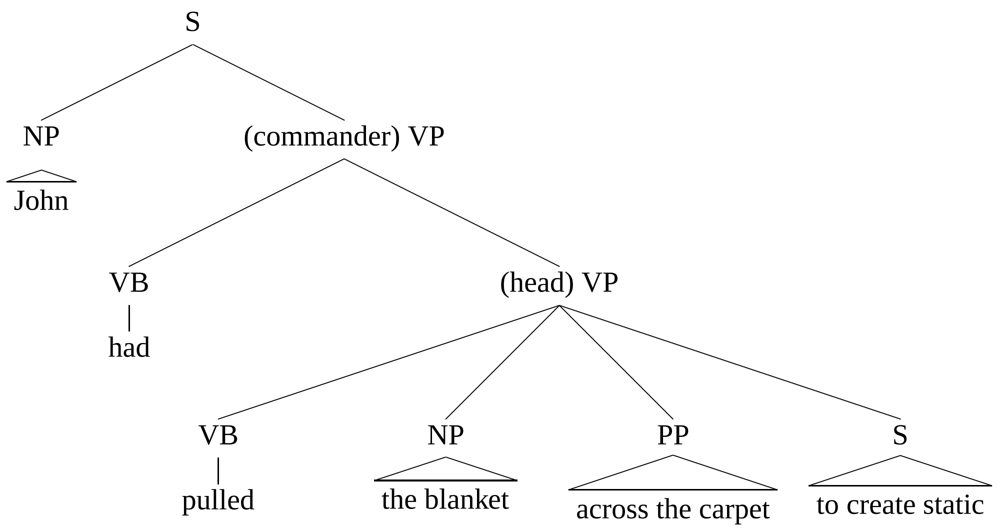


Figure 1: Example parse tree for w="pulled", from which is extracted the syntactic features: *morph=normal subj dobj sent-comp subj=john dobj=blanket prep=across across-obj=carpet.*

- If there is a PP complement, what is the preposition, and what is the object of the preposition?

Semantic features:

- What is the Named Entity tag (PERSON, ORGANIZATION, LOCATION, UNKNOWN) for each proper noun in the syntactic positions above?

- What are the possible WordNet synsets and hypernyms for each noun in the syntactic positions above? (Nouns are not explicitly disambiguated; all possible synsets and hypernyms for the noun are included.)

This set of local features relies on access to syntactic structure as well as semantic class information, and attempts to model richer linguistic information about predicate arguments. However, the heuristics for extracting the syntactic features are able to identify subjects and objects of only simple clauses. The heuristics also do not differentiate between arguments and adjuncts; for example, the feature *sent-comp* is intended to identify clausal complements such as in (S (NP Mary) (VP (VB called) (S him a bastard))), but Figure 1 shows how a purpose clause can be mistakenly labeled as a clausal complement.

2.3 Evaluation

We tested the system on the 1806 test instances of the 29 verbs from the English lexical sample task for Senseval-2 (Palmer et al., 2001). Accuracy was defined to be the fraction of the instances for which the system got the correct sense. All significance testing between different accuracies was done using a one-tailed *z*-test, assuming a binomial distribution of the successes; differences in accuracy were considered to be significant if $p <= 0.050$.

In Senseval-2, senses involving multi-word constructions could be identified directly from the sense tags themselves, and the head word and satellites of multi-word constructions were explicitly marked in the training and test data. We trained one model for each of the verbs and used a filter to consider only phrasal senses whenever there were satellites of multi-word constructions marked in the test data.

Feature	Accuracy
co	0.571
co+syn	0.598
co+syn+sem	0.625

Table 1: Accuracy of system on Senseval-2 verbs using topical features and different subsets of local features.

Table 1 shows the accuracy of the system using topical features and different subsets of local features. Adding features from richer linguistic sources always improves accuracy. Adding lexical syntactic ("syn") features improves accuracy significantly over using just collocational ("co") features ($p = 0.050$). When semantic class ("sem") features are added, the improvement is also significant.

Adding topical information to all the local features improves accuracy, but not significantly; when the topical features are removed the accuracy of our system falls only slightly, to 62.0%. Senses based on domain or topic occur rarely in the Senseval-2 corpus. Most of the information provided by topical features already seem to be captured by the local features for the frequent senses.

Features	Accuracy
co+syn	0.598
co+syn+ne	0.597
co+syn+wn	0.623
co+syn+ne+wn	0.625

Table 2: Accuracy of system on Senseval-2 verbs, using topical features and different subsets of semantic class features.

Semantic class information plays a significant role in sense distinctions. Table 2 shows the relative contribution of adding only named entity tags to the collocational and syntactic features ("co+syn+ne"), versus adding only the WordNet classes ("co+syn+wn"), versus adding both named entity and WordNet classes ("co+syn+ne+wn"). Adding all possible WordNet noun class features for arguments contributes a large number of parameters to the model, but this use of WordNet with no separate disambiguation of noun arguments proves to be very useful. In fact, the use of WordNet for common nouns proves to be even more beneficial than the use of a named entity tagger for proper nouns. Given enough data, the maximum entropy model is able to assign high weights to the correct hypernyms of the correct noun sense if they represent defining selectional restrictions.

Incorporating topical keywords as well as collocational, syntactic, and semantic local features, our system achieves 62.5% accuracy. This is in comparison to the 61.1% accuracy achieved by (Lee and Ng, 2002), which has been the best published result on this corpus.

3 PropBank semantic annotations

Our WSD system uses heuristics to attempt to detect predicate arguments from parsed sentences. However, recognition of predicate argument structures is not straightforward, because a natural language will have several different syntactic realizations of the same predicate argument relations.

PropBank is a corpus in which verbs are annotated with semantic tags, including coarse-grained sense distinctions and predicate-argument structures. PropBank adds a layer of semantic annotation to the Penn Wall Street Journal Treebank II. An important goal is to provide consistent predicate-argument structures across different syntactic realizations of the same verb. Polysemous verbs are also annotated with different *framesets*. Frameset tags are based on differences in subcategorization frames and correspond to a coarse notion of word senses.

A verb's semantic arguments in PropBank are numbered beginning with 0. Arg0 is roughly equivalent to the thematic role of Agent, and Arg1 usually corresponds to Theme or Patient; however, argument labels are not necessarily consistent across different senses of the same verb, or across different verbs, as thematic roles are usually taken to be. In addition to the core, numbered arguments, verbs can take any of a set of general, adjunct-like arguments (ARGM), whose labels are derived from the Treebank functional tags (DIRection, LOCation, etc.).

PropBank provides manual annotation of predicate-argument information for a large number of verb instances in the Senseval-2 data set. The intersection of PropBank and Senseval-2 forms a corpus containing gold-standard annotations of fine-grained WordNet senses, coarse-grained PropBank framesets, and PropBank role labels. The combination of such gold-standard semantic annotations provides a unique opportunity to investigate the role of predicate-argument features in word sense disambiguation, for both coarse-grained framesets and fine-grained WordNet senses.

3.1 PropBank features

We conducted experiments on the effect of using features from PropBank for sense-tagging verbs. Both PropBank role labels and PropBank framesets were used. In the case of role labels, only the gold-standard labels found in PropBank were used, because the best automatic semantic role labelers only perform at about 84% precision and 75% recall (Pradhan et al., 2004).

From the PropBank annotation for each sentence, we extracted the following features:

1. Labels of the semantic roles: rel, ARG0, ARG1, ARG2-WITH, ARG2, ..., ARGM-LOC, ARGM-TMP, ARGM-NEG, ...

2. Syntactic labels of the constituent instantiating each semantic role: ARG0=NP, ARGM-TMP=PP, ARG2-WITH=PP, ...

3. Head word of each constituent in (2): rel=called, sats=up, ARG0=company, ARGM-TMP=day, ...

4. Semantic classes (named entity tag, WordNet hypernyms) of the nouns in (3): ARGOsyn=ORGANIZATION, ARGOsyn=16185, ARGM-TMPsyn=13018, ...

When a numbered role appears in a prepositional phrase (e.g., ARG2-WITH), we take the "head word" to be the object of the preposition. If a constituent instantiating some semantic role is a trace, we take the head of its referent instead.

- [$_{\mathrm{ARG0}}$ Mr. Bush] has [$_{\mathrm{rel}}$ called] [$_{\mathrm{ARG1-for}}$ for an agreement by next September at the latest] .

For example, the PropBank features that we extract for the sentence above are:
arg0 arg0=bush arg0syn=person arg0syn=1740 ...
rel rel=called
arg1-for arg1 arg1=agreement arg1syn=12865 ...

3.2 Role labels for frameset tagging

We collected all instances of the Senseval-2 verbs from the PropBank corpus. Only 20 of these verbs had more than one frameset in the PropBank corpus, resulting in 4887 instances of polysemous verbs. The instances for each word were partitioned randomly into 10 equal parts, and the system was tested on each part after being trained on the remaining nine. For these 20 verbs with more than one PropBank frameset tag, choosing the most frequent frameset gives a baseline accuracy of 76.0%.

The sentences were automatically pos-tagged with the Ratnaparki tagger and parsed with the Collins parser. We extracted local contextual features as for WordNet sense-tagging and used the local features to train our WSD system on the coarse-grained sense-tagging task of automatically assigning PropBank frameset tags. We tested the effect of using only collocational features ("co") for frameset tagging, as well as using only PropBank role features ("pb") or only our original syntactic/semantic features ("synsem") for this task, and found that the combination of collocational features with PropBank features worked best. The system has the worst performance on the word *strike*, which has a high number of framesets and a low number of training instances. Table 3 shows the performance of the system on different subsets of local features.

Feature	Accuracy
baseline	0.760
co	0.853
synsem	0.859
co+synsem	0.883
pb	0.901
co+pb	0.908
co+synsem+pb	0.907

Table 3: Accuracy of system on frameset-tagging task for verbs with more than one frameset, using different types of local features (no topical features); all features except pb were extracted from automatically pos-tagged and parsed sentences.

We obtained an overall accuracy of 88.3% using our original local contextual features. However, the system's performance improved significantly when we used only PropBank role features, achieving an accuracy of 90.1%. Furthermore, adding collocational features and heuristically extracted syntactic/semantic features to the PropBank features do not provide additional information and affects the accuracy of frameset-tagging only negligibly. It is not surprising that for the coarse-grained sense-tagging task of assigning the correct PropBank frameset tag to a verb, using the PropBank role labels is better than syntactic/semantic features heuristically extracted from parses because these heuristics are meant to capture the predicate-argument information that is encoded more directly in the PropBank role labels.

Even when the original local features were extracted from the gold-standard pos-tagged and parsed sentences of the Penn Treebank, the system performed significantly worse than when PropBank role features were used. This suggests that more effort should be applied to improving the heuristics for extracting syntactic features.

We also experimented with adding topical features and ARGM features from PropBank. In all cases, these additional features reduced overall accuracy, but the difference was never significant ($p >= 0.100$). Topical features do not help because frameset tags are based on differences in subcategorization frames and not on the domain or topic. ARGM features do not help because they are supposedly used uniformly across verbs and framesets.

3.3 Role labels for WordNet sense-tagging

We experimented with using PropBank role labels for fine-grained WordNet sense-tagging. While ARGM features are not useful for coarse-grained frameset-tagging, some sense distinctions in WordNet are based on adverbial modifiers, such as "live well" or "serves someone well." Therefore, we included PropBank ARGM features in our models for WordNet sense-tagging to capture a wider range of linguistic behavior. We looked at the 2571 instances of 29 Senseval-2 verbs that were in both Senseval-2 and the PropBank corpus.

Features	Accuracy
co	0.628
synsem	0.638
co+synsem	0.666
pb	0.656
co+pb	0.681
co+synsem+pb	0.694

Table 4: Accuracy of system on WordNet sense-tagging for instances in both Senseval-2 and PropBank, using different types of local features (no topical features).

Table 4 shows the accuracy of the system on WordNet sense-tagging using different subsets of features; all features except pb were extracted from automatically pos-tagged and parsed sentences. By adding PropBank role features to our original local feature set, accuracy rose from 0.666 to to 0.694 on this subset of the Senseval-2 verbs ($p = 0.020$); the extraction of syntactic features from the parsed sentences is again not successfully capturing all the predicate-argument information that is explicit in PropBank.

The verb "match" illustrates why accuracy improves using additional PropBank features. As shown in Figure 2, the matched objects may occur in different grammatical relations with respect to the verb (subject, direct object, object of a preposition), but they each have an ARG1 semantic role label in PropBank.[3] Furthermore, only one of the matched objects needs to be specified, as in Example 3 where the second matched object (presumably the company's prices) is unstated. Our heuristics do not handle these alternations, and cannot detect that the syntactic subject in Example 1 has a different semantic role than the subject of Example 3.

Roleset **match.01** "match":
Arg0: person performing match
Arg1: matching objects
Ex1: [$_{ARG1}$ the wallpaper] [$_{rel}$ matched] [$_{ARG1}$ the paint]
Ex2: [$_{ARG0}$ The architect] [$_{rel}$ matched] [$_{ARG1}$ the paint] [$_{Arg1-WITH}$ with the wallpaper]
Ex3: [$_{ARG0}$ The company] [$_{rel}$ matched] [$_{ARG1}$ Kodak's higher prices]

Figure 2: PropBank roleset for "match"

Our basic WSD system (using local features extracted from automatic parses) confused WordNet Sense 1 with Sense 4:

1. match, fit, correspond, check, jibe, gibe, tally, agree – (be compatible, similar or consistent; coincide in their characteristics; "The two stories don't agree in many details"; "The handwriting checks with the signature on the check"; "The suspect's fingerprints don't match those on the gun")

4. equal, touch, rival, match – (be equal to in

[3]PropBank annotation for "match" allows multiple ARG1 labels, one for each of the matching objects. Other verbs that have more than a single ARG1 in PropBank include: "attach, bolt, coincide, connect, differ, fit, link, lock, pin, tack, tie."

quality or ability; “Nothing can rival cotton for durability”; “Your performance doesn’t even touch that of your colleagues”; “Her persistence and ambition only matches that of her parents”)

The senses are differentiated in that the matching objects (ARG1) in Sense 4 have some quantifiable characteristic that can be measured on some scale, whereas those in Sense 1 are more general. Gold-standard PropBank annotation of ARG1 allows the system to generalize over the semantic classes of the arguments and distinguish these two senses more accurately.

3.4 Frameset tags for WordNet sense-tagging

PropBank frameset tags (either gold-standard or automatically tagged) were incorporated as features in our WSD system to see if knowing the coarse-grained sense tags would be useful in assigning fine-grained WordNet sense tags. A frameset tag for the instance was appended to each feature; this effectively partitions the feature set according to the coarse-grained sense provided by the frameset. To automatically tag an instance of a verb with its frameset, the set of all instances of the verb in PropBank was partitioned into 10 subsets, and an instance in one subset was tagged by training a maximum entropy model on the instances in the other nine subsets. Various local features were considered, and the same feature types were used to train the frameset tagger and the WordNet sense tagger that used the automatically-assigned frameset.

For the 20 Senseval-2 verbs that had more than one frameset in PropBank, we extracted all instances that were in both Senseval-2 and PropBank, yielding 1468 instances. We examined the effect of incorporating the gold-standard PropBank frameset tags into our maximum entropy models for these 20 verbs by partitioning the instances according to their frameset tag. Table 5 shows a breakdown of the accuracy by feature type. Adding the gold-standard frameset tag (“*fset”) to our original local features (“orig”) did not increase the accuracy significantly. However, the increase in accuracy (from 59.7% to 62.8%) was significant when these frameset tags were incorporated into the model that used both our original features and all the PropBank features.

Feature	Accuracy
orig	0.564
orig*fset	0.587
orig+pb	0.597
(orig+pb)*fset	0.628

Table 5: Accuracy of system on WordNet sense-tagging of 20 Senseval-2 verbs with more than one frameset, with and without gold-standard frameset tag.

However, partitioning the instances using the automatically generated frameset tags has no significant effect on the system’s performance; the information provided by the automatically assigned coarse-grained sense tag is already encoded in the features used for fine-grained sense-tagging.

4 Related Work

Our approach of using rich linguistic features combined in a single maximum entropy framework contrasts with that of (Florian et al., 2002). Their feature space was much like ours, but did not include semantic class features for noun complements. With this more impoverished feature set, they experimented with combining diverse classifiers to achieve an improvement of 2.1% over all parts of speech (noun, verb, adjective) in the Senseval-2 lexical sample task; however, this improvement was over an initial accuracy of 56.6% on verbs, indicating that their performance is still below ours for verbs.

(Lee and Ng, 2002) explored the relative contribution of different knowledge sources and learning algorithms to WSD; they used Support Vector Machines (SVM) and included local collocations and syntactic relations, and also found that adding syntactic features improved accuracy. Our features are similar to theirs, but we added semantic class features for the verb arguments. We found that the difference in machine learning algorithms did not play a large role in performance; when we used our features in SVM we obtained almost no difference in performance over using maximum entropy models with Gaussian priors.

(Gomez, 2001) described an algorithm using WordNet to simultaneously determine verb senses and attachments of prepositional phrases, and identify thematic roles and adjuncts; our work is different in that it is trained on *manually* annotated corpora to show the relevance of semantic roles for verb sense disambiguation.

5 Conclusion

We have shown that disambiguation of verb senses can be improved by leveraging information about predicate arguments and their semantic classes. Our system performs at the best published accuracy on the English verbs of Senseval-2 even though our heuristics for extracting syntactic features fail to identify all and only the arguments of a verb. We show that associating WordNet semantic classes with nouns is beneficial even without explicit disambiguation of the noun senses because, given enough data, maximum entropy models are able to assign high weights to the correct hypernyms of the correct noun sense if they represent defining selectional restrictions. Knowledge of gold-standard predicate-argument information from PropBank improves WSD on both coarse-grained senses (PropBank framesets) and fine-grained WordNet senses. Furthermore, partitioning instances according to their gold-standard frameset tags, which are based on differences in subcategorization frames, also improves the system's accuracy on fine-grained WordNet sense-tagging. Our experiments suggest that sense disambiguation for verbs can be improved through more accurate extraction of features representing information such as that contained in the framesets and predicate argument structures annotated in PropBank.

6 Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This paper describes research that was conducted while the first author was at the University of Pennsylvania.

References

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: A high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, DC.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain, July.

Radu Florian, Silviu Cucerzan, Charles Schafer, and David Yarowsky. 2002. Combining classifiers for word sense disambiguation. *Natural Language Engineering*, 8(4):327–341.

Fernando Gomez. 2001. An algorithm for aspects of semantic interpretation using an enhanced wordnet. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Jane Grimshaw. 1990. *Argument Structure*. MIT Press, Cambridge, MA.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of Third International Conference on Language Resources and Evaluation*, Las Palmas, Canary Islands, Spain, May.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Philadelphia, PA.

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

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Martha Palmer, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang. 2001. English tasks: All-words and verb lexical sample. In *Proceedings of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems*, Toulouse, France, July.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics*, May.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

D. Yarowsky. 1995. *Three Machine Learning Algorithms for Lexical Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania Department of Computer and Information Sciences.

Aggregation improves learning: experiments in natural language generation for intelligent tutoring systems

Barbara Di Eugenio and **Davide Fossati** and **Dan Yu**
University of Illinois
Chicago, IL, 60607, USA
{bdieugen,dfossa1,dyu6}@uic.edu

Susan Haller
University of Wisconsin - Parkside
Kenosha, WI 53141, USA
haller@cs.uic.edu

Michael Glass
Valparaiso University
Valparaiso, IN, 46383, USA
Michael.Glass@valpo.edu

Abstract

To improve the interaction between students and an intelligent tutoring system, we developed two Natural Language generators, that we systematically evaluated in a three way comparison that included the original system as well. We found that the generator which intuitively produces the best language does engender the most learning. Specifically, it appears that *functional* aggregation is responsible for the improvement.

1 Introduction

The work we present in this paper addresses three issues: evaluation of Natural Language Generation (NLG) systems, the place of aggregation in NLG, and NL interfaces for Intelligent Tutoring Systems.

NLG systems have been evaluated in various ways, such as via task efficacy measures, i.e., measuring how well the users of the system perform on the task at hand (Young, 1999; Carenini and Moore, 2000; Reiter et al., 2003). We also employed task efficacy, as we evaluated the learning that occurs in students interacting with an Intelligent Tutoring System (ITS) enhanced with NLG capabilities. We focused on sentence planning, and specifically, on aggregation. We developed two different feedback generation engines, that we systematically evaluated in a three way comparison that included the original system as well. Our work is novel for NLG evaluation in that we focus on one specific component of the NLG process, aggregation. Aggregation pertains to combining two or more of the messages to be communicated into one sentence (Reiter and Dale, 2000). Whereas it is considered an essential task of an NLG system, its specific contributions to the effectiveness of the text that is eventually produced have rarely been assessed (Harvey and Carberry, 1998). We found that syntactic aggregation does not improve learning, but that what we call *functional* aggregation does. Further, we ran a controlled data collection in order to provide a more solid empirical base for aggregation rules than what is normally found in the literature, e.g. (Dalianis, 1996; Shaw, 2002).

As regards NL interfaces for ITSs, research on the next generation of ITSs (Evens et al., 1993; Litman et al., 2004; Graesser et al., 2005) explores NL as one of the keys to bridging the gap between current ITSs and human tutors. However, it is still not known whether the NL interaction between students and an ITS does in fact improve learning. We are among the first to show that this is the case.

We will first discuss DIAG, the ITS shell we are using, and the two feedback generators that we developed, *DIAG-NLP1* and *DIAG-NLP2*. Since the latter is based on a corpus study, we will briefly describe that as well. We will then discuss the formal evaluation we conducted and our results.

2 Natural Language Generation for DIAG

DIAG (Towne, 1997) is a shell to build ITSs based on interactive graphical models that teach students to troubleshoot complex systems such as home heating and circuitry. A DIAG application presents a student with a series of troubleshooting problems of increasing difficulty. The student tests *indicators* and tries to infer which faulty part (RU) may cause the abnormal states detected via the indicator readings. RU stands for *replaceable unit*, because the only course of action for the student to fix the problem is to replace faulty components in the graphical simulation.

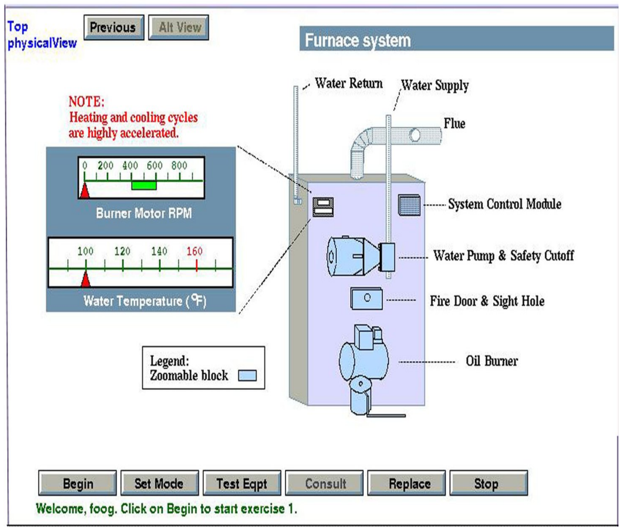


Figure 1: The furnace system

Fig. 1 shows the furnace, one subsystem of the home heating system in our DIAG application. Fig. 1 includes indicators such as the gauge labeled Water Temperature, RUs, and complex modules (e.g., the Oil Burner) that contain indicators and RUs. Complex components are zoomable.

At any point, the student can consult the tutor via the Consult menu (cf. the Consult button in Fig. 1). There are two main types of queries: *ConsultInd(icator)* and *ConsultRU*. *ConsultInd* queries are used mainly when an indicator shows an abnormal reading, to obtain a hint regarding which RUs may cause the problem. DIAG discusses the RUs that should be most suspected given the symptoms the student has already observed. *ConsultRU* queries are mainly used to obtain feedback on the diagnosis that a certain RU is faulty. DIAG responds with an assessment of that diagnosis and provides evidence for it in terms of the symptoms that have been observed relative to that RU.

The original DIAG system (*DIAG-orig*) uses very simple templates to assemble the text to present to the student. The top parts of Figs. 2 and 3 show the replies provided by *DIAG-orig* to a *ConsultInd* on the *Visual Combustion Check*, and to a *ConsultRu* on the *Water Pump*.

The highly repetitive feedback by *DIAG-orig* screams for improvements based on aggregation techniques. Our goal in developing *DIAG-NLP1* and *DIAG-NLP2* was to assess whether simple, rapidly deployable NLG techniques would lead to measurable improvements in the student's learning. Thus, in both cases it is still DIAG that performs content determination, and provides to *DIAG-NLP1* and *DIAG-NLP2* a file in which the facts to be communicated are written – a *fact* is the basic unit of information that underlies each of the clauses in a reply by *DIAG-orig*. The only way we altered the interaction between student and system is the actual language that is presented in the output window. In *DIAG-NLP1* we mostly explored using syntactic aggregation to improve the feedback, whereas *DIAG-NLP2* is corpus-based and focuses on functional aggregation. In both *DIAG-NLP1* and *DIAG-NLP2*, we use EXEMPLARS (White and Caldwell, 1998), an object-oriented, rule-based generator. The rules (called *exemplars*) are meant to capture an exemplary way of achieving a communicative goal in a given context. EXEMPLARS selects rules by traversing the exemplar specialization hierarchy and evaluating the applicability conditions associated with each exemplar.

The visual combustion check is igniting which is abnormal (normal is combusting).
Oil Nozzle always
    produces this abnormality when it fails.
Oil Supply Valve always
    produces this abnormality when it fails.
Oil pump always
    produces this abnormality when it fails.
Oil Filter always
    produces this abnormality when it fails.
System Control Module sometimes
    produces this abnormality when it fails.
Ignitor Assembly never
    produces this abnormality when it fails.
Burner Motor always
    produces this abnormality when it fails.

The visual combustion check indicator is igniting.
This is abnormal.
Normal is combusting.

Within the furnace system,
    this is sometimes caused if
        the System Control Module has failed.

Within the Oil Burner
    this is never caused if
        the Ignitor Assembly has failed.
    In contrast, this is always caused if
        the Burner Motor, Oil Filter, Oil Pump,
        Oil Supply Valve, or Oil Nozzle has failed.

The combustion is abnormal.
In the oil burner, check the units along the path of the oil and the burner motor.

Figure 2: Answers to *ConsultInd* by *DIAG-orig*, *DIAG-NLP1* and *DIAG-NLP2*

```
Water pump is a very poor suspect.
Some symptoms you have seen conflict with that theory.
Water pump sound was normal.
This normal indication never results when this unit fails.
Visual combustion check was igniting.
This abnormal indication never results when this unit fails.
Burner Motor RMP Gauge was 525.
This normal indication never results when this unit fails.
```

```
The Water pump is a very poor suspect.
Some symptoms you have seen conflict with that theory.

The following indicators never display normally
when this unit fails.
Within the furnace system,
    the Burner Motor RMP Gauge is 525.
Within the water pump and safety cutoff valve,
    the water pump sound indicator is normal.

The following indicators never display abnormally
when this unit fails.
Within the fire door sight hole,
    the visual combustion check indicator is igniting.
```

```
The water pump is a poor suspect since the water pump
sound is ok.
You have seen that the combustion is abnormal.
Check the units along the path of the oil and the electrical
devices.
```

Figure 3: Answers to *ConsultRu* by *DIAG-orig*, *DIAG-NLP1* and *DIAG-NLP2*

2.1 *DIAG-NLP1*: Syntactic aggregation

DIAG-NLP1[1] (i) introduces syntactic aggregation (Dalianis, 1996; Huang and Fiedler, 1996; Reape and Mellish, 1998; Shaw, 2002) and what we call *structural* aggregation, namely, grouping parts according to the structure of the system; (ii) generates some referring expressions; (iii) models a few rhetorical relations; and (iv) improves the format of the output.

The middle parts of Figs. 2 and 3 show the revised output produced by *DIAG-NLP1*. E.g., in Fig. 2 the RUs of interest are grouped by the system modules that contain them (Oil Burner and Furnace System), and by the likelihood that a certain RU causes the observed symptoms. In contrast to the original answer, the revised answer highlights that the *Ignitor Assembly* cannot cause the symptom.

In *DIAG-NLP1*, EXEMPLARS accesses the SNePS Knowledge Representation and Reasoning System (Shapiro, 2000) for static domain information.[2] SNePS makes it easy to recognize structural similarities and use shared structures. Using SNePS, we can examine the dimensional structure of an aggregation and its values to give preference to aggregations with top-level dimensions that have fewer values, to give summary statements when a dimension has many values that are reported on, and to introduce simple text structuring in terms of rhetorical relations, inserting relations like *contrast* and *concession* to highlight distinctions between dimensional values (see Fig. 2, middle).

DIAG-NLP1 uses the GNOME algorithm (Kibble and Power, 2000) to generate referential expressions. Importantly, using SNePS propositions can be treated as discourse entities, added to the discourse model and referred to (see *This is ... caused if ...* in Fig. 2, middle). Information about lexical realization, and choice of referring expression is encoded in the appropriate exemplars.

2.2 *DIAG-NLP2*: functional aggregation

In the interest of rapid prototyping, *DIAG-NLP1* was implemented without the benefit of a corpus study. *DIAG-NLP2* is the empirically grounded version of the feedback generator. We collected 23 tutoring interactions between a student using the DIAG tutor on home heating and two human tutors, for a total of 272 tutor turns, of which 235 in reply to *ConsultRU* and 37 in reply to *ConsultInd* (the type of student query is automatically logged). The tutor and the student are in different rooms, sharing images of the same DIAG tutoring screen. When the student consults DIAG, the tutor sees, in tabular form, the information that DIAG would use in generating its advice — the same "fact file" that DIAG gives to *DIAG-NLP1* and *DIAG-NLP2* — and types a response that substitutes for DIAG's. The tutor is presented with this information because we wanted to uncover empirical evidence for aggregation rules in our domain. Although we cannot constrain the tutor to mention only the facts that DIAG would have communicated, we can analyze how the tutor uses the information provided by DIAG.

We developed a coding scheme (Glass et al., 2002) and annotated the data. As the annotation was performed by a single coder, we lack measures of intercoder reliability. Thus, what follows should be taken as observations rather than as rigorous findings – useful observations they clearly are, since

[1]*DIAG-NLP1* actually augments and refines the first feedback generator we created for DIAG, *DIAG-NLP0* (Di Eugenio et al., 2002). *DIAG-NLP0* only covered (i) and (iv).

[2]In DIAG, domain knowledge is hidden and hardly accessible. Thus, in both *DIAG-NLP1* and *DIAG-NLP2* we had to build a small knowledge base that contains domain knowledge.

DIAG-NLP2 is based on these observations and its language fosters the most learning.

Our coding scheme focuses on four areas. Fig. 4 shows examples of some of the tags (the SCM is the System Control Module). Each tag has from one to five additional attributes (not shown) that need to be annotated too.

Domain ontology. We tag objects in the domain with their class *indicator, RU* and their states, denoted by *indication* and *operationality*, respectively.
Tutoring actions. They include (i) *Judgment*. The tutor evaluates what the student did. (ii) *Problem solving*. The tutor suggests the next course of action. (iii) The tutor imparts *Domain Knowledge*.
Aggregation. Objects may be *functional aggregates*, such as *the oil burner*, which is a system component that includes other components; *linguistic aggregates*, which include plurals and conjunctions; or a *summary* over several unspecified indicators or RUs. *Functional/linguistic aggregate* and *summary* tags often co-occur, as shown in Fig. 4.
Relation to DIAG's output. Contrary to all other tags, in this case we annotate the input that DIAG gave the tutor. We tag its portions as *included / excluded / contradicted*, according to how it has been dealt with by the tutor.

Tutors provide explicit problem solving directions in 73% of the replies, and evaluate the student's action in 45% of the replies (clearly, they do both in 28% of the replies, as in Fig. 4). As expected, they are much more concise than DIAG, e.g., they never mention RUs that cannot or are not as likely to cause a certain problem, such as, respectively, the *ignitor assembly* and *the SCM* in Fig. 2.

As regards aggregation, 101 out of 551 RUs, i.e. 18%, are labelled as summary; 38 out of 193 indicators, i.e. 20%, are labelled as summary. These percentages, though seemingly low, represent a considerable amount of aggregation, since in our domain some items have very little in common with others, and hence cannot be aggregated. Further, tutors aggregate parts functionally rather than syntactically. For example, the same assemblage of parts, i.e., oil nozzle, supply valve, pump, filter, etc., can be described as *the other items on the fuel line* or as *the path of the oil flow*.

Finally, *directness* – an attribute on the *indicator* tag – encodes whether the tutor explicitly talks about the indicator (e.g., *The water temperature gauge reading is low*), or implicitly via the object to which the indicator refers (e.g., *the water is too cold*). 110 out of 193 indicators, i.e. 57%, are marked as *implicit*, 45, i.e. 41%, as *explicit*, and 2% are not marked for directness (the coder was free to leave attributes unmarked). This, and the 137 occurrences of *indication*, prompted us to refer to objects and their states, rather than to indicators (as implemented by Steps 2 in Fig. 5, and 2(b)i, 3(b)i, 3(c)i in Fig. 6, which generate *The combustion is abnormal* and *The water pump sound is OK* in Figs. 2 and 3).

2.3 Feedback Generation in *DIAG-NLP2*

In *DIAG-NLP1* the fact file provided by DIAG is directly processed by EXEMPLARS. In contrast, in *DIAG-NLP2* a planning module manipulates the information before passing it to EXEMPLARS. This module decides which information to include according to the type of query the system is responding to, and produces one or more *Sentence Structure* objects. These are then passed to EXEMPLARS that transforms them into Deep Syntactic Structures. Then, a sentence realizer, RealPro (Lavoie and Rambow, 1997), transforms them into English sentences.

Figs. 5 and 6 show the control flow in *DIAG-NLP2* for feedback generation for *ConsultInd* and *ConsultRU*. Step 3a in Fig. 5 chooses, among all the RUs that DIAG would talk about, only those that would definitely result in the observed symptom. Step 2 in the AGGREGATE procedure in Fig. 5 uses a simple heuristic to decide whether and how to use functional aggregation. For each RU, its possible aggregators and the number n of units it covers are listed in a table (e.g., *electrical devices* covers 4 RUs, *ignitor, photoelectric cell, transformer* and *burner motor*). If a group of REL-RUs contains k units that a certain aggregator *Agg* covers, if $k < \frac{n}{2}$, *Agg* will not be used; if $\frac{n}{2} \leq k < n$, *Agg* preceded by *some of* will be used; if $k = n$, *Agg* will be used.

DIAG-NLP2 does not use SNePS, but a relational database storing relations, such as the ISA hierarchy (e.g., *burner motor* IS-A RU), information about referents of indicators (e.g., *room temperature gauge* REFERS-TO *room*), and correlations between RUs and the indicators they affect.

3 Evaluation

Our empirical evaluation is a three group, between-subject study: one group interacts with *DIAG-orig*,

$[_{judgment}$ $[_{replaceable-unit}$ the ignitor] is a poor suspect] since $[_{indication}$ combustion is working] during startup. The problem is that the SCM is shutting the system off during heating.
$[_{domain-knowledge}$ The SCM reads $[_{summary}$ $[_{linguistic-aggregate}$ input signals from sensors]] and uses the signals to determine how to control the system.]
$[_{problem-solving}$ Check the sensors.]

Figure 4: Examples of a coded tutor reply

```
1. IND ← queried indicator
2. Mention the referent of IND and its state
3. IF IND reads abnormal,
   (a) REL-RUs ← choose relevant RUs
   (b) AGGR-RUs ← AGGREGATE(REL-RUs)
   (c) Suggest to check AGGR-RUs

AGGREGATE(RUs)
1. Partition REL-RUs into subsets by system structure
2. Apply functional aggregation to subsets
```

Figure 5: *DIAG-NLP2*: Feedback generation for *ConsultInd*

```
1. RU ← queried RU
   REL-IND ← indicator associated to RU

2. IF RU warrants suspicion,
   (a) state RU is a suspect
   (b) IF student knows that REL-IND is abnormal
       i. remind him of referent of REL-IND and
          its abnormal state
       ii. suggest to replace RU
   (c) ELSE suggest to check REL-IND

3. ELSE
   (a) state RU is not a suspect
   (b) IF student knows that REL-IND is normal
       i. use referent of REL-IND and its normal state
          to justify judgment
   (c) IF student knows of abnormal indicators OTHER-INDs
       i. remind him of referents of OTHER-INDs
          and their abnormal states
       ii. FOR each OTHER-IND
           A. REL-RUs ← RUs associated with OTHER-IND
           B. AGGR-RUs ← AGGREGATE(REL-RUs)
              ∪ AGGR-RUs
       iii. Suggest to check AGGR-RUs
```

Figure 6: *DIAG-NLP2*: Feedback generation for *ConsultRU*

one with *DIAG-NLP1*, one with *DIAG-NLP2*. The 75 subjects (25 per group) were all science or engineering majors affiliated with our university. Each subject read some short material about home heating, went through one trial problem, then continued through the curriculum on his/her own. The curriculum consisted of three problems of increasing difficulty. As there was no time limit, every student solved every problem. Reading materials and curriculum were identical in the three conditions.

While a subject was interacting with the system, a log was collected including, for each problem: whether the problem was solved; total time, and time spent reading feedback; how many and which indicators and RUs the subject consults DIAG about; how many, and which RUs the subject replaces. We will refer to all the measures that were automatically collected as *performance measures*.

At the end of the experiment, each subject was administered a questionnaire divided into three parts. The first part (the posttest) consists of three questions and tests what the student learned about the domain. The second part concerns whether subjects remember their actions, specifically, the RUs they replaced. We quantify the subjects' recollections in terms of precision and recall with respect to the log that the system collects. We expect precision and recall of the replaced RUs to correlate with *transfer*, namely, to predict how well a subject is able to apply what s/he learnt about diagnosing malfunctions to new problems. The third part concerns usability, to be discussed below.

We found that subjects who used *DIAG-NLP2* had significantly higher scores on the posttest, and were significantly more correct (higher precision) in remembering what they did. As regards performance measures, there are no so clear cut results. As regards usability, subjects prefer *DIAG-NLP1*/2 to *DIAG-orig*, however results are mixed as regards which of the two they actually prefer.

In the tables that follow, boldface indicates significant differences, as determined by an analysis of variance performed via ANOVA, followed by post-hoc Tukey tests.

Table 1 reports learning measures, average across the three problems. *DIAG-NLP2* is significantly better as regards PostTest score ($F = 10.359, p = 0.000$), and RU Precision ($F = 4.719, p = 0.012$). Performance on individual questions in the

	DIAG-orig	*DIAG-NLP1*	*DIAG-NLP2*
PostTest	0.72	0.69	**0.90**
RU Precision	0.78	0.70	**0.91**
RU Recall	.53	.47	.40

Table 1: Learning Scores

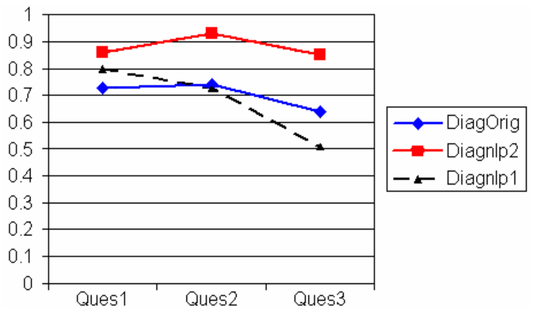


Figure 7: Scores on PostTest questions

PostTest[3] is illustrated in Fig. 7. Scores in *DIAG-NLP2* are always higher, significantly so on questions 2 and 3 ($F = 8.481, p = 0.000$, and $F = 7.909, p = 0.001$), and marginally so on question 1 ($F = 2.774, p = 0.069$).[4]

	D-Orig	*D-NLP1*	*D-NLP2*
Total Time	30'17"	28'34"	34'53"
RU Replacements	8.88	11.12	11.36
ConsultInd	22.16	**6.92**	28.16
Avg. Reading Time	8"	14"	**2"**
ConsultRU	63.52	45.68	52.12
Avg. Reading Time	5"	4"	5"

Table 2: Performance Measures

Table 2 reports performance measures, cumulative across the three problems, other than average reading times. Subjects don't differ significantly in the time they spend solving the problems, or in the number of *RU replacements* they perform. DIAG's assumption (known to the subjects) is that there is only one broken RU per problem, but the simulation allows subjects to replace as many as they want without any penalty before they come to the correct solution. The trend on *RU replacements* is opposite what we would have hoped for: when repairing a real system, replacing parts that are working should clearly be kept to a minimum, and subjects replace fewer parts in *DIAG-orig*.

The next four entries in Table 2 report the number of queries that subjects ask, and the average time it takes subjects to read the feedback. The subjects ask significantly fewer *ConsultInd* in *DIAG-NLP1* ($F = 8.905, p = 0.000$), and take significantly less time reading *ConsultInd* feedback in *DIAG-NLP2* ($F = 15.266, p = 0.000$). The latter result is not surprising, since the feedback in *DIAG-NLP2* is much shorter than in *DIAG-orig* and *DIAG-NLP1*. Neither the reason not the significance of subjects asking many fewer *ConsultInd* of *DIAG-NLP1* are apparent to us – it happens for *ConsultRU* as well, to a lesser, not significant degree.

We also collected usability measures. Although these are not usually reported in ITS evaluations, in a real setting students should be more willing to sit down with a system that they perceive as more friendly and usable. Subjects rate the system along four dimensions on a five point scale: clarity, usefulness, repetitiveness, and whether it ever misled them (the scale is appropriately arranged: the highest clarity but the lowest repetitiveness receive 5 points). There are no significant differences on individual dimensions. Cumulatively, *DIAG-NLP2* (at 15.08) slightly outperforms the other two (*DIAG-orig* at 14.68 and *DIAG-NLP1* at 14.32), however, the difference is not significant (highest possible rating is 20 points).

	prefer	neutral	disprefer
DIAG-NLP1 to *DIAG-orig*	28	5	17
DIAG-NLP2 to *DIAG-orig*	34	1	15
DIAG-NLP2 to *DIAG-NLP1*	24	1	25

Table 3: User preferences among the three systems

	prefer	neutral	disprefer
Consult Ind.	8	1	16
Consult RU	16	0	9

Table 4: *DIAG-NLP2* versus *DIAG-NLP1*

	natural	concise	clear	contentful
DIAG-NLP1	4	8	10	23
DIAG-NLP2	16	8	11	12

Table 5: Reasons for system preference

Finally,[5] on paper, subjects compare two pairs of versions of feedback: in each pair, the first feedback

[3]The three questions are: 1. Describe the main subsystems of the furnace. 2. What is the purpose of (a) the oil pump (b) the system control module? 3. Assume the photoelectric cell is covered with enough soot that it could not detect combustion. What impact would this have on the system?

[4]The PostTest was scored by one of the authors, following written guidelines.

[5]Subjects can also add free-form comments. Only few did

is generated by the system they just worked with, the second is generated by one of the other two systems. Subjects say which version they prefer, and why (they can judge the system along one or more of four dimensions: natural, concise, clear, contentful). The first two lines in Table 3 show that subjects prefer the NLP systems to *DIAG-orig* (marginally significant, $\chi^2 = 9.49, p < 0.1$). *DIAG-NLP1* and *DIAG-NLP2* receive the same number of preferences; however, a more detailed analysis (Table 4) shows that subjects prefer *DIAG-NLP1* for feedback to *ConsultInd*, but *DIAG-NLP2* for feedback to *ConsultRu* (marginally significant, $\chi^2 = 5.6, p < 0.1$). Finally, subjects find *DIAG-NLP2* more natural, but *DIAG-NLP1* more contentful (Table 5, $\chi^2 = 10.66, p < 0.025$).

4 Discussion and conclusions

Our work touches on three issues: aggregation, evaluation of NLG systems, and the role of NL interfaces for ITSs.

In much work on aggregation (Huang and Fiedler, 1996; Horacek, 2002), aggregation rules and heuristics are shown to be plausible, but are not based on any hard evidence. Even where corpus work is used (Dalianis, 1996; Harvey and Carberry, 1998; Shaw, 2002), the results are not completely convincing because we do not know for certain the content to be communicated from which these texts supposedly have been aggregated. Therefore, positing empirically based rules is guesswork at best. Our data collection attempts at providing a more solid empirical base for aggregation rules; we found that tutors exclude significant amounts of factual information, and use high degrees of aggregation based on functionality. As a consequence, while part of our rules implement standard types of aggregation, such as conjunction via shared participants, we also introduced functional aggregation (see *conceptual* aggregation (Reape and Mellish, 1998)).

As regards evaluation, NLG systems have been evaluated e.g. by using human judges to assess the quality of the texts produced (Coch, 1996; Lester and Porter, 1997; Harvey and Carberry, 1998); by comparing the system's performance to that of humans (Yeh and Mellish, 1997); or through task efficacy measures, i.e., measuring how well the users of the system perform on the task at hand (Young, 1999; Carenini and Moore, 2000; Reiter et al., 2003). The latter kind of studies generally contrast different interventions, i.e. a baseline that does not use NLG and one or more variations obtained by parameterizing the NLG system. However, the evaluation does not focus on a specific component of the NLG process, as we did here for aggregation.

Regarding the role of NL interfaces for ITSs, only very recently have the first few results become available, to show that first of all, students do learn when interacting in NL with an ITS (Litman et al., 2004; Graesser et al., 2005). However, there are very few studies like ours, that evaluate specific features of the NL interaction, e.g. see (Litman et al., 2004). In our case, we did find that different features of the NL feedback impact learning. Although we contend that this effect is due to functional aggregation, the feedback in *DIAG-NLP2* changed along other dimensions, mainly using referents of indicators instead of indicators, and being more strongly directive in suggesting what to do next. Of course, we cannot argue that our best NL generator is equivalent to a human tutor – e.g., dividing the number of *ConsultRU* and *ConsultInd* reported in Sec. 2.2 by the number of dialogues shows that students ask about 10 *ConsultRus* and 1.5 *ConsultInd* per dialogue when interacting with a human, many fewer than those they pose to the ITSs (cf. Table 2) (regrettably we did not administer a PostTest to students in the human data collection). We further discuss the implications of our results for NL interfaces for ITSs in a companion paper (Di Eugenio et al., 2005).

The DIAG project has come to a close. We are satisfied that we demonstrated that even not overly sophisticated NL feedback can make a difference; however, the fact that *DIAG-NLP2* has the best language and engenders the most learning prompts us to explore more complex language interactions. We are pursuing new exciting directions in a new domain, that of basic data structures and algorithms. We are investigating what distinguishes expert from novice tutors, and we will implement our findings in an ITS that tutors in this domain.

Acknowledgments. This work is supported by the Office of Naval Research (awards N00014-99-1-0930 and N00014-00-1-0640), and in part by the National Science Foundation (award IIS 0133123). We are grateful to CoGenTex Inc. for making EXEMPLARS and RealPro available to us.

so, and the distribution of topics and of evaluations is too broad to be telling.

References

Giuseppe Carenini and Johanna D. Moore. 2000. An empirical study of the influence of argument conciseness on argument effectiveness. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong.

José Coch. 1996. Evaluating and comparing three text-production techniques. In *COLING96, Proceedings of the Sixteenth International Conference on Computational Linguistics*, pages 249–254

Hercules Dalianis. 1996. *Concise Natural Language Generation from Formal Specifications*. Ph.D. thesis, Department of Computer and Systems Science, Stocholm University. Technical Report 96-008.

Barbara Di Eugenio, Michael Glass, and Michael J. Trolio. 2002. The DIAG experiments: Natural Language Generation for Intelligent Tutoring Systems. In *INLG02, The Third International Natural Language Generation Conference*, pages 120–127.

Barbara Di Eugenio, Davide Fossati, Dan Yu, Susan Haller, and Michael Glass. 2005. Natural language generation for intelligent tutoring systems: a case study. In *AIED 2005, the 12th International Conference on Artificial Intelligence in Education*.

M. W. Evens, J. Spitkovsky, P. Boyle, J. A. Michael, and A. A. Rovick. 1993. Synthesizing tutorial dialogues. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 137–140.

Michael Glass, Heena Raval, Barbara Di Eugenio, and Maarika Traat. 2002. The DIAG-NLP dialogues: coding manual. Technical Report UIC-CS 02-03, University of Illinois - Chicago.

A.C. Graesser, N. Person, Z. Lu, M.G. Jeon, and B. McDaniel. 2005. Learning while holding a conversation with a computer. In L. PytlikZillig, M. Bodvarsson, and R. Brunin, editors, *Technology-based education: Bringing researchers and practitioners together*. Information Age Publishing.

Terrence Harvey and Sandra Carberry. 1998. Integrating text plans for conciseness and coherence. In *ACL/COLING 98, Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 512–518.

Helmut Horacek. 2002. Aggregation with strong regularities and alternatives. In *International Conference on Natural Language Generation*.

Xiaoron Huang and Armin Fiedler. 1996. Paraphrasing and aggregating argumentative text using text structure. In *Proceedings of the 8th International Workshop on Natural Language Generation*, pages 21–30.

Rodger Kibble and Richard Power. 2000. Nominal generation in GNOME and ICONOCLAST. Technical report, Information Technology Research Institute, University of Brighton, Brighton, UK.

Benoît Lavoie and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*.

James C. Lester and Bruce W. Porter. 1997. Developing and empirically evaluating robust explanation generators: the KNIGHT experiments. *Computational Linguistics*, 23(1):65–102.

D. J. Litman, C. P. Rosé, K. Forbes-Riley, K. VanLehn, D. Bhembe, and S. Silliman. 2004. Spoken versus typed human and computer dialogue tutoring. In *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems*, Maceio, Brazil.

Mike Reape and Chris Mellish. 1998. Just what *is* aggregation anyway? In *Proceedings of the European Workshop on Natural Language Generation*.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press.

Ehud Reiter, Roma Robertson, and Liesl Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144:41–58.

S. C. Shapiro. 2000. SNePS: A logic for natural language understanding and commonsense reasoning. In L. M. Iwanska and S. C. Shapiro, editors, *Natural Language Processing and Knowledge Representation*. AAAI Press/MIT Press.

James Shaw. 2002. A corpus-based analysis for the ordering of clause aggregation operators. In *COLING02, Proceedings of the 19th International Conference on Computational Linguistics*.

Douglas M. Towne. 1997. Approximate reasoning techniques for intelligent diagnostic instruction. *International Journal of Artificial Intelligence in Education*.

Michael White and Ted Caldwell. 1998. Exemplars: A practical, extensible framework for dynamic text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 266–275, Niagara-on-the-Lake, Canada.

Ching-Long Yeh and Chris Mellish. 1997. An empirical study on the generation of anaphora in Chinese. *Computational Linguistics*, 23(1):169–190.

R. Michael Young. 1999. Using Grice's maxim of quantity to select the content of plan descriptions. *Artificial Intelligence*, 115:215–256.

Empirically-based Control of Natural Language Generation

Daniel S. Paiva
Department of Informatics
University of Sussex
Brighton, UK
danielpa@sussex.ac.uk

Roger Evans
Information Technology Research Institute
University of Brighton
Brighton, UK
Roger.Evans@itri.brighton.ac.uk

Abstract

In this paper we present a new approach to controlling the behaviour of a natural language generation system by correlating internal decisions taken during free generation of a wide range of texts with the surface stylistic characteristics of the resulting outputs, and using the correlation to control the generator. This contrasts with the generate-and-test architecture adopted by most previous empirically-based generation approaches, offering a more efficient, generic and holistic method of generator control. We illustrate the approach by describing a system in which stylistic variation (in the sense of Biber (1988)) can be effectively controlled during the generation of short medical information texts.

1 Introduction

This paper[1] is concerned with the problem of controlling the output of natural language generation (NLG) systems. In many application scenarios the generator's task is underspecified, resulting in multiple possible solutions (texts expressing the desired content), all equally good to the generator, but not equally appropriate for the application. Customising the generator directly to overcome this generally leads to ad-hoc, non-reusable solutions. A more modular approach is a generate-and-test architecture, in which all solutions are generated, and then ranked or otherwise selected according to their appropriateness in a separate post-process. Such architectures have been particularly prominent in the recent development of empirically-based approaches to NLG, where generator outputs can be selected according to application requirements acquired directly from human subjects (e.g. Walker *et al.* (2002)) or statistically from a corpus (e.g. Langkilde-Geary (2002)). However, this approach suffers from a number of drawbacks:

1. It requires generation of all, or at least many solutions (often hundreds of thousands), expensive both in time and space, and liable to lead to unnecessary interactions with other components (e.g. knowledge bases) in complex systems. Recent advances in the use of packed representations ameliorate some of these issues, but the basic need to compare a large number of solutions in order to rank them remains.
2. The 'test' component generally does not give fine-grained control — for example, in a statistically-based system it typically measures how close a text is to some single notion of ideal (actually, statistically average) output.
3. Use of an external filter does not combine well with any control mechanisms *within* the generator: e.g. controlling combinatorial explosion of modifier attachment or adjective order.

In this paper we present an empirically-based method for controlling a generator which overcomes these deficiencies. It controls the generator internally, so that it can produce just one (locally) optimal solution; it employs a model of language *variation*, so that the generator can be controlled within a multidimensional space of possible variants; its view of the generator is completely holistic, so that it can accommodate any other control mechanisms intrinsic to the generation task.

[1] Paiva and Evans (2004) provides an overview of our framework and detailed comparison with previous approaches to stylistic control (like Hovy (1988), Green and DiMarco (1993) and Langkilde-Geary (2002)). This paper provides a more detailed account of the system and reports additional experimental results.

Proceedings of the 43rd Annual Meeting of the ACL, pages 58–65,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

To illustrate our approach we describe a system for controlling 'style' in the sense of Biber (1988) during the generation of short texts giving instructions about doses of medicine. The paper continues as follows. In §2 we describe our overall approach. We then present the implemented system (§3) and report on our experimental evaluation (§4). We end with a discussion of conclusions and future directions (§5).

2 Overview of the Approach

Our overall approach has two phases: (1) offline calculation of the control parameters, and (2) online application to generation. In the first phase we determine a set of *correlation equations*, which capture the relationship between surface linguistic features of generated texts and the internal generator decisions that gave rise to those texts (see figure 1). In the second phase, these correlations are used to guide the generator to produce texts with particular surface feature characteristics (see figure 2).

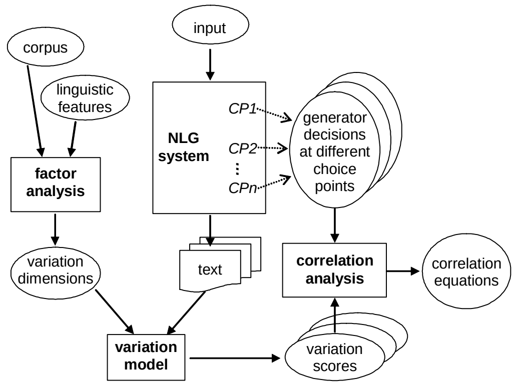


Figure 1: Offline processing

The starting point is a corpus of texts which represents all the variability that we wish to capture. Counts for (surface) linguistic features from the texts in the corpus are obtained, and a factor analysis is used to establish dimensions of variation in terms of these counts: each dimension is defined by a weighted sum of scores for particular features, and factor analysis determines the combination that best accounts for the variability across the whole corpus. This provides a *language variation model* which can be used to score a new text along each of the identified dimensions, that is, to locate the text in the variation space determined by the corpus.

The next step is to take a generator which can generate across the range of variation in the corpus, and identify within it the key choice points (CP_1, CP_2, … CP_n) in its generation of a text. We then allow the generator to freely generate all possible texts from one or more inputs. For each text so generated we record (a) the text's score according to the variation model and (b) the set of decisions made at each of the selected choice points in the generator. Finally, for a random sample of the generated texts, a statistical correlation analysis is undertaken between the scores and the corresponding generator decisions, resulting in correlation equations which predict likely variation scores from generator decisions.

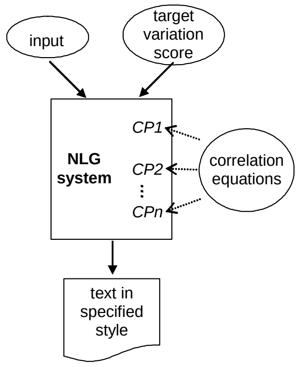


Figure 2: Online processing

In the second phase, the generator is adapted to use the correlation equations to conduct a best-first search of the generation space. As well as the usual input, the generator is supplied with target scores for each dimension of variation. At each choice point, the correlation equations are used to predict which choice is most likely to move closer to the target score for the final text.

This basic architecture makes no commitment to what is meant by 'variation', 'linguistic features', 'generator choice points', or even 'NLG system'. The key ideas are that a statistical analysis of surface features of a corpus of texts can be used to define a model of variation; this model can then be used to control a generator; and the model can also be used to evaluate the generator's performance. In the next section we describe a concrete instantiation of this architecture, in which 'variation' is stylistic variation as characterised by a collection of shallow lexical and syntactic features.

3 An Implemented System

In order to evaluate the effectiveness of this general approach, we implemented a system which attempts to control style of text generated as defined by Biber (1988) in short text (typically 2-3 sentences) describing medicine dosage instructions.

3.1 Factor Analysis

Biber characterised style in terms of very shallow linguistic features, such as presence of pronouns, auxiliaries, passives etc. By using factor analysis techniques he was able to determine complex correlations between the occurrence and non-occurrence of such features in text, which he used to characterise different styles of text.[2]

We adopted the same basic methodology, applied to a smaller more consistent corpus of just over 300 texts taken from proprietary patient information leaflets. Starting with around 70 surface linguistic features as variables, our factor analysis yielded two main factors (each containing linguistic features grouped in positive and negative correlated subgroups) which we used as our dimensions of variation. We interpreted these dimensions as follows (this is a subjective process — factor analysis does not itself provide any interpretation of factors): dimension 1 ranges from texts that try to involve the reader (high positive score) to text that try to be distant from the reader (high negative score); dimension 2 ranges from texts with more pronominal reference and a higher proportion of certain verbal forms (high positive score) to text that use full nominal reference (high negative score).[3]

3.2 Generator Architecture

The generator was constructed from a mixture of existing components and new implementation, using a fairly standard overall architecture as shown in figure 3. Here, dotted lines show the control flow and the straight lines show data flow — the choice point annotations are described below.

The *input constructor* takes an input specification and, using a background database of medicine information, creates a network of concepts and relations (see figure 4) using a schema-based approach (McKeown, 1985).

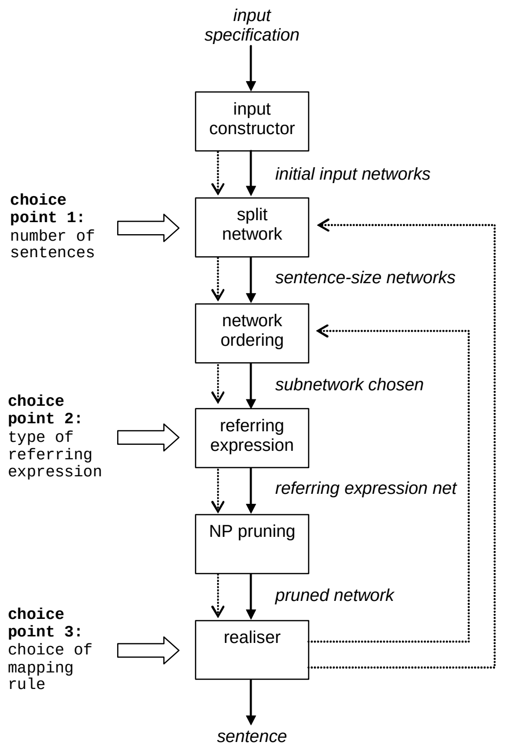


Figure 3: Generator architecture with choice points

Each network is then split into subnetworks by the *split network* module. This partitions the network by locating 'proposition' objects (marked with a double-lined box in figure 4) which have no parent and tracing the subnetwork reachable from each one. We call these subnetworks *propnets*. In figure 4, there are two propnets, rooted in [1:take] and [9:state] — proposition [15:state] is not a root as it can be reached from [1:take]. A list of all possible groupings of these propnets is obtained[4], and one of the possible combinations is passed to the network ordering module. This is the first source of non-determinism in our system, marked as *choice point one* in figure 3. A combination of subnetworks will be material for the realisation of one paragraph and each subnetwork will be realised as one sentence.

[2] Some authors (e.g. Lee (1999)) have criticised Biber for making assumptions about the validity and generalisability of his approach to English language as a whole. Here, however, we use his methodology to characterise whatever variation exists without needing to make any broader claims.

[3] Full details of the factor analysis can be found in (Paiva 2000).

[4] For instance, with three propnets (A, B and C) the list of combinations would be [(A,B,C), (A,BC), (AB, C), (AC,B), (ABC)].

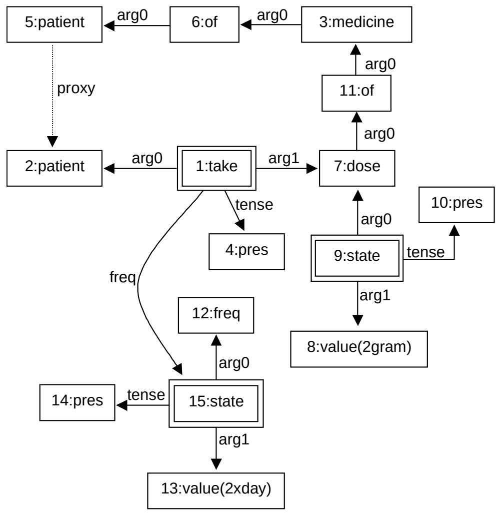


Figure 4: Example of semantic network produced by the input constructor[5]

The *network ordering* module receives a combination of subnetworks and orders them based on the number of common elements between each subnetwork. The strategy is to try to maximise the possibility of having a smooth transition from one sentence to the next in accordance with Centering Theory (Grosz *et al*., 1995), and so increase the possibility of having a pronoun generated.

The *referring expression* module receives one subnetwork at a time and decides, for each object that is of type [thing], which type of referring expression will be generated. The module is re-used from the Riches system (Cahill *et al*., 2001) and it generates either a definite description or a pronoun. This is the second source of non-determinism in our system, marked as *choice point two* in figure 3. Referring expression decisions are recorded by introducing additional nodes into the network, as shown for example in figure 5 (a fragment of the network in figure 4, with the additional nodes).

NP pruning is responsible for erasing from a referring expression subnetwork all the nodes that can be transitively reached from a node marked to be pronominalised. This prevents the realiser from trying to express the information twice. In figure 5, [7:dose] is marked to be pronominalised, so the concepts [11:of] and [3:medicine] do not need to be realised, so they are pruned.

[5] Although some of the labels in this figure look like words, they bear no direct relation to words in the surface text — for example, 'of' may be realised as a genitive construction or a possessive.

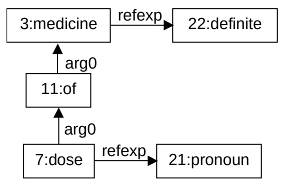


Figure 5: Referring expressions and pruning

The *realiser* is a re-implementation of Nicolov's (1999) generator, extended to use the wide-coverage lexicalised grammar developed in the LEXSYS project (Carroll *et al*., 2000), with further semantic extensions for the present system. It selects grammar rules by matching their semantic patterns to subnetworks of the input, and tries to generate a sentence consuming the whole input. In general there are several rules linking each piece of semantics to its possible realisation, so this is our third, and most prolific, source of non-determinism in the architecture, marked as *choice point three* in figure 3.

A few examples of outputs for the input represented in figure 4 are:

> the dose of the patient 's medicine is taken twice a day. it is two grams.
>
> the two-gram dose of the patient 's medicine is taken twice a day.
>
> the patient takes the two-gram dose of the patient 's medicine twice a day.

From a typical input corresponding to 2-3 sentences, this generator will generate over a 1000 different texts.

3.3 Tracing Generator Behaviour

In order to control the generator's behaviour we first allow it to run freely, recording a 'trace' of the decisions it makes at each choice point during the production of each text. Although there are only three choice points in figure 3, the control structure included two loops: an outer loop which ranges over the sequence of propnets, generating a sentence for each one, and an inner loop which ranges over subnetworks of a propnet as realisation rules are chosen. So the decision structure for even a small text may be quite complex.

In the experiments reported here, the trace of the generation process is simply a record of the number of times each decision (choice point, and what choice was made) occurred. Paiva (2004) discusses more complex tracing models, where the context of each decision (for example, what the preceding decision was) is recorded and used in the correlation. However the best results were obtained using just the simple decision-counting model (perhaps in part due to data sparseness for more complex models).

3.4 Correlating Decisions with Text Features

By allowing the generator to freely generate all possible output from a single input, we recorded a set of <trace, text> pairs ranging across the full variation space. From these pairs we derived corresponding <decision-count, factor-score> pairs, to which we applied a very simple correlational technique, *multivariate linear regression analysis*, which is used to find an estimator function for a linear relationship (i.e., one that can be approximated by a straight line) from the data available for several variables (Weisberg, 1985). In our case we want to predict the value for a score in a stylistic dimension (SS_i) based on a configuration of generator decisions (GD_j) as seen in equation 1.

(eq. 1) $SS_i = x_0 + x_1GD_1 + \ldots + x_nGD_n + \varepsilon$ [6]

We used three randomly sampled data sets of 1400, 1400 and 5000 observations obtained from a potential base of about 1,400,000 different texts that could be produced by our generator from a single input. With each sample, we obtained a regression equation for each stylistic dimension separately. In the next subsections we will present the final results for each of the dimensions separately.

Regression on Stylistic Dimension 1

For the regression model on the first stylistic dimension (SS1), the generator decisions that were used in the regression analysis[7] are: imperative with one object sentences (IMP_VNP), V_NP_PP agentless passive sentences (PAS_VNPP), V_NP by-passives (BYPAS_VN), and N_PP clauses (NPP) and these are all decisions that happen in the realiser, i.e., at the third choice point in the architecture. This resulted in the regression equation shown in equation 2.

(eq. 2)

$$SS1 = 6.459 - (1.460{*}\text{NPP}) - (1.273{*}\text{BYPAS\_VN}) - (1.826{*}\text{PAS\_VNPP}) + (1.200{*}\text{IMP\_VNP})$$ [8]

The coefficients for the regression on SS1 are unstandardised coefficients, i.e. the ones that are used when dealing with raw counts for the generator decisions.

The coefficient of determination (R^2), which measures the proportion of the variance of the dependent variable about its mean that is explained by the independent variables, had a reasonably high value (.895)[9] and the analysis of variance obtained an *F* test of 1701.495.

One of the assumptions that this technique assumes is the linearity of the relation between the dependent and the independent variables (i.e., in our case, between the stylistic scores in a dimension and the generator decisions). The analysis of the residuals resulted in a graph that had some problems but that resembled a normal graph (see (Paiva, 2004) for more details).

Regression on Stylistic Dimension 2

For the regression model on the second stylistic dimension (SS2) the variables that we used were: the number of times a network was split (SPLITNET), generation of a pronoun (RE_PRON), auxiliary verb (VAUX), noun with determiner (NOUN), transitive verb (VNP), and agentless passive (PAS_VNP) — the first type of decision happens in the split network module (our first choice point); the second, in the referring expression module (second choice point); and the rest in the realiser (third choice point).

The main results for this model are as follows: the coefficient of determination (R^2) was .959 and the analysis of variance obtained an *F* test of 2298.519. The unstandardised regression coefficients for this model can be seen in eq. 3.

(eq. 3)

$$SS2 = -27.208 - (1.530{*}\text{VNP}) + (2.002{*}\text{RE\_PRON}) - (.547{*}\text{NOUN}) + (.356{*}\text{VAUX}) + (.860{*}\text{SPLITNET}) + (.213{*}\text{PAS\_VNP})$$ [10]

[6] SS_i represents a stylistic score and is the *dependent variable* or *criterion* in the regression analysis; the GD_j's represent generator decisions and are called the *independent variables* or *predictors*; the x_j's are weights, and ε is the error.

[7] The process of determining the regression takes care of eliminating the variables (i.e. generator decisions) that are not useful to estimate the stylistic dimensions.

[8] This specific equation came from the sample with 5,000 observations — the equations obtained from the other samples are very similar to this one.

[9] All the statistical results presented in this paper are significant at the 0.01 level (two-tailed).

[10] This specific equation comes from one of the samples of 1,400 observations.

With this second model we did not find any problems with the linearity assumptions as the analysis of the residuals gave a normal graph.

4 Controlling the Generator

These regression equations characterise the way in which generator decisions influence the final style of the text (as measured by the stylistic factors). In order to control the generator, the user specifies a target stylistic score for each dimension of the text to be generated. At each choice point during generation, all possible decisions are collected in a list and the regression equations are used to order them. The equations allow us to estimate the subsequent values of SS1 and SS2 for each of the possible decisions, and the decisions are ordered according to the distance of the resulting scores from the target scores — the closer the score, the better the decision.

Hence the search algorithm that we are using here is the *best-first search*, i.e., the best local solution according to an evaluation function (which in this case is the Euclidian distance from the target and the resulted value obtained by using the regression equation) is tried first but all the other local solutions are kept in order so backtracking is possible.

In this paper we report on tests of two internal aspects of the system[11]. First we wish to know how good the generator is at hitting a user-specified target — i.e., how close are the scores given by the regression equations for the first text generated to the user's input target scores. Second, we wish to know how good the regression equation scores are at modelling the original stylistic factors — i.e., we want to compare the regression scores of an output text with the factor analysis scores. We address these questions across the whole of the two-dimensional stylistic space, by specifying a rectangular grid of scores spanning the whole space, and asking the generator to produce texts for each grid point from the same semantic input specification.

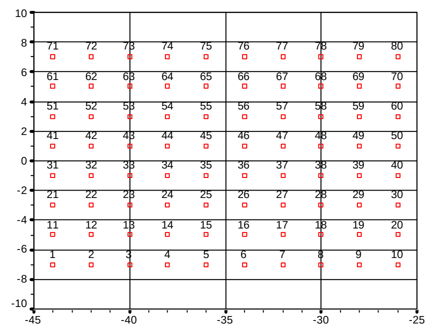


Figure 6: Target scores for the texts

In this case we divided the scoring space with an 8 by 10 grid pattern as shown in figure 6.[12] Each point specifies the target scores for each text that should be generated (the number next to each point is an identifier of each text). For instance, text number 1 was targeted at coordinate (−7, −44), whereas text number 79 was targeted at coordinate (+7, −28).

4.1 Comparing Target Points and Regression Scores

In the first part of this experiment we wanted to know how close to the user-specified target coordinates the resulting regression scores of the first generated text were. This can be done in two different ways. The first is to plot the resulting regression scores (see figure 7) and visually check if it mirrors the grid-shape pattern of the target points (figure 6) — this can be done by inspecting the text identifiers[13]. This can be a bit misleading because there will always be variation around the target point that was supposed to be achieved (i.e., there is a margin for error) and this can blur the comparison unfavourably.

[11] We are not dealing with external (user) evaluation of the system and of the stylistic dimensions we obtained — this was left for future work. Nonetheless, Sigley (1997) showed that the dimensions obtained with factor analysis and people's perception have a high correlation.

[12] The range for each scale comes from the maximum and minimum values for the factors obtained in the samples of generated texts.

[13] Note that some texts obtained the same regression score and, in the statistical package, only one was numbered. Those instances are: 1 and 7; 18 and 24; 22 and 28.

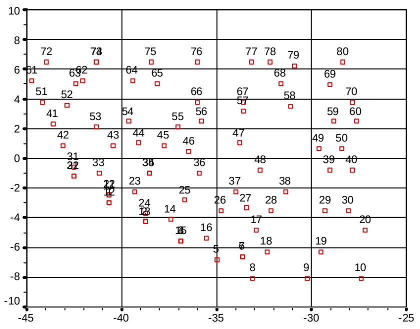


Figure 7: Texts scored by using the regression equation

A more formal comparison can be made by plotting the target points versus the regression results for each dimension separately and obtaining a correlation measure between these values. These correlations are shown in figure 8 for SS1 (left) and SS2 (right). The degree of correlation (R^2) between the values of target and regression points is 0.9574 for SS1 and 0.942 for SS2, which means that the search mechanism is working very satisfactorily on both dimensions.[14]

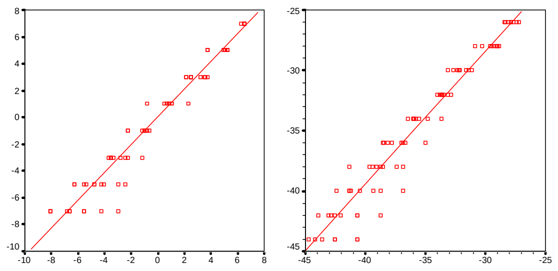


Figure 8: Plotting target points versus regression results on SS1 (left) and SS2 (right)

4.2 Comparing Target Points and Stylistic Scores

In the second part of this experiment we wanted to know whether the regression equations were doing the job they were supposed to do by comparing the regression scores with stylistic scores obtained (from the factor analysis) for each of the generated texts. In figure 9 we plotted the texts in a graph in accordance with their stylistic scores (once again, some texts occupy the same point so they do not appear).

[14] All the correlational figures (R^2) presented for this experiment are significant at the 0.01 level (two-tailed).

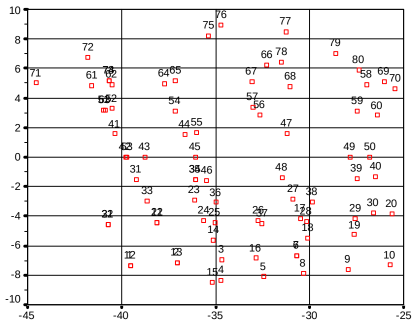


Figure 9: Texts scored using the two stylistic dimension obtained in our factor analysis

In the ideal situation, the generator would have produced texts with the perfect regression scores and they would be identical to the stylistic scores, so the graph in the figure 9 would be like a grid-shape one as in figure 6. However we have already seen in figure 7, that this is not the case for the relation between the target coordinates and the regession scores. So we did not expect the plot of stylistic scores 1 (SS1) against stylistic scores 2 (SS2) to be a perfect grid.

Figure 10 (left-hand side) shows the relation between the target points and the scores obtained from the original factor equation of SS1. The value of R^2, which represents their correlation, is high (0.9458), considering that this represents the possible accumulation of errors of two stages: from the target to the regression scores, and then from the regression to the actual factor scores. On the right of figure 10 we can see the plotting of the target points and their respective factor scores on SS2. The correlation obtained is also reasonably high (R^2 = 0.9109).

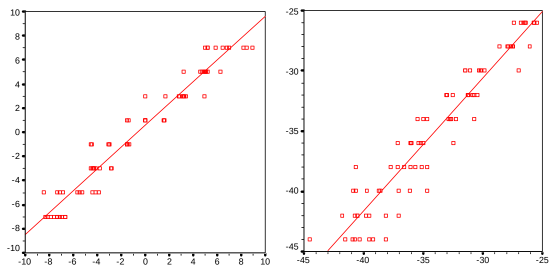


Figure 10: Plotting target points versus factor scores on SS1 (left) and SS2 (right)

5 Discussion and Future Work

These results demonstrate that it is possible to provide effective control of a generator correlating internal generator behaviour with characteristics of the resulting texts. It is important to note that these two sets of variables (generator decision and surface features) are in principle quite independent of each other. Although in some cases there are strong correlations (for example, the generator's use of a 'passive' rule, correlates with the occurrence of passive participles in the text), in others the relationship is much less direct (for example, the choice of how many subnetworks to split a network into, i.e., SPLITNET, does not correspond to any feature in the factor analysis), and the way individual features combine into significant factors may be quite different.

Another feature of our approach is that we do not assume some pre-defined notion of parameters of variation – variation is characterised completely by a corpus (in contrast to approaches which use a corpus to characterise a *single* style). The disadvantage of this is that variation is not grounded in some 'intuitive' notion of style: the interpretation of the stylistic dimensions is subjective and tentative. However, as no comprehensive computationally realisable theory of style yet exists, we believe that this approach has considerable promise for practical, empirically-based stylistic control.

The results reported here also make us think that a possible avenue for future work is to explore the issue of what types of problems the generalisation induced by our framework (which will be discussed below) can be applied to. This paper dealt with an application to stylistic variation but, in theory, the approach can be applied to any kind of process to which there is a sorting function that can impose an order, using *a measurable scale* (e.g., ranking), onto the outputs of another process.

Schematically the approach can be abstracted to any sort of problem of the form shown in figure 11. Here there is a *producer* process outputting a large number of solutions. There is also a *sorter* process which will classify those solutions in a certain order. The numerical value associated with the output by the *sorter* can be correlated with the decisions the producer took to generate the output. The same correlation and control mechanism used in this paper can be introduced in the producer process, making it controllable with respect to the sorting dimension.

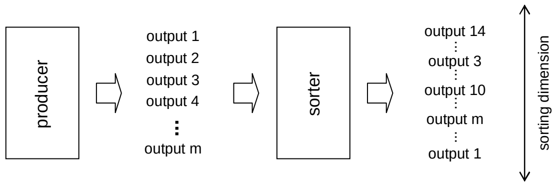


Figure 11: The producer-sorter scheme.

References

Biber, Douglas (1988) *Variation across speech and writing*. Cambridge University Press.

Cahill, Lynne; J. Carroll; R. Evans; D. Paiva; R. Power; D. Scott; and K. van Deemter From RAGS to RICHES: exploiting the potential of a flexible generation architecture. *Proceedings of ACL/EACL 2001*, pp. 98-105.

Carroll, John; N. Nicolov; O. Shaumyan; M. Smets; and D. Weir (2000) Engineering a wide-coverage lexicalized grammar. *Proceedings of the Fifth International Workshop on Tree Adjoining Grammars and Related Frameworks.*

Green, Stephen J.; and C. DiMarco (1993) Stylistic decision-making in NLG. In *Proceedings of the 4th European Workshop on Natural Language Generation*. Pisa, Italy.

Grosz, Barbara J.; A.K. Joshi; and S. Weinstein (1995) *Centering: A Framework for Modelling the Local Coherence of Discourse*. Institute for Research in Cognitive Science, IRCS-95-01, University of Pennsylvania.

Hovy, Eduard H. (1988) *Generating natural language under pragmatic constraints*. Lawrence Erlbaum Associates.

Langkilde-Geary, Irene. (2002) An empirical verification of coverage and correctness for a general-purpose sentence generator. *Proceeding of INLG'02*, pp. 17-24.

Lee, David (1999) *Modelling Variation in Spoken And Written English: the Multi-Dimensional Approach Revisited*. PhD thesis, University of Lancaster, UK.

McKeown, Kathleen R. (1985) *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press.

Nicolov, Nicolas (1999) *Approximate Text Generation from Non-hierarchical Representations in a Declarative Framework*. PhD Thesis, University of Edinburgh.

Paiva, Daniel S. (2000) Investigating style in a corpus of pharmaceutical leaflets: results of a factor analysis. *Proceedings of the Student Workshop of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'2000),* Hong Kong, China.

Paiva, Daniel S. (2004) *Using Stylistic Parameters to Control a Natural Language Generation System*. PhD Thesis, University of Brighton, Brighton, UK.

Paiva, Daniel S.; R. Evans (2004) A Framework for Stylistically Controlled Generation. In *Proceedings of the 3rd International Conference on Natural Language Generation (INLG'04)*. New Forest, UK.

Sigley, Robert (1997) Text categories and where you can stick them: a crude formality index. *International Journal of Corpus Linguistics*, volume 2, number 2, pp. 199-237.

Walker, Marilyn; O. Rambow, and M. Rogati (2002) Training a Sentence Planner for Spoken Dialogue Using Boosting. *Computer Speech and Language, Special Issue on Spoken Language Generation*. July.

Weisberg, Sanford (1985) *Applied Linear Regression*, 2nd edition. John Wiley & Sons.

Towards Developing Generation Algorithms for Text-to-Text Applications

Radu Soricut and Daniel Marcu
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
{radu, marcu}@isi.edu

Abstract

We describe a new sentence realization framework for text-to-text applications. This framework uses IDL-expressions as a representation formalism, and a generation mechanism based on algorithms for intersecting IDL-expressions with probabilistic language models. We present both theoretical and empirical results concerning the correctness and efficiency of these algorithms.

1 Introduction

Many of today's most popular natural language applications – Machine Translation, Summarization, Question Answering – are text-to-text applications. That is, they produce textual outputs from inputs that are also textual. Because these applications need to produce well-formed text, it would appear natural that they are the favorite testbed for generic generation components developed within the Natural Language Generation (NLG) community. Over the years, several proposals of generic NLG systems have been made: Penman (Matthiessen and Bateman, 1991), FUF (Elhadad, 1991), Nitrogen (Knight and Hatzivassiloglou, 1995), Fergus (Bangalore and Rambow, 2000), HALogen (Langkilde-Geary, 2002), Amalgam (Corston-Oliver et al., 2002), etc. Instead of relying on such generic NLG systems, however, most of the current text-to-text applications use other means to address the generation need. In Machine Translation, for example, sentences are produced using application-specific "decoders", inspired by work on speech recognition (Brown et al., 1993), whereas in Summarization, summaries are produced as either extracts or using task-specific strategies (Barzilay, 2003). The main reason for which text-to-text applications do not usually involve generic NLG systems is that such applications do not have access to the kind of information that the input representation formalisms of current NLG systems require. A machine translation or summarization system does not usually have access to deep subject-verb or verb-object relations (such as ACTOR, AGENT, PATIENT, POSSESSOR, etc.) as needed by Penman or FUF, or even shallower syntactic relations (such as `subject`, `object`, `premod`, etc.) as needed by HALogen.

In this paper, following the recent proposal made by Nederhof and Satta (2004), we argue for the use of IDL-expressions as an application-independent, information-slim representation language for text-to-text natural language generation. IDL-expressions are created from strings using four operators: concatenation ($\cdot$), interleave ($\|$), disjunction ($\vee$), and lock ($\times$). We claim that the IDL formalism is appropriate for text-to-text generation, as it encodes meaning only via words and phrases, combined using a set of formally defined operators. Appropriate words and phrases can be, and usually are, produced by the applications mentioned above. The IDL operators have been specifically designed to handle natural constraints such as word choice and precedence, constructions such as phrasal combination, and underspecifications such as free word order.

NLG System	Representation (formalism)	Representation (computational)	Generation (mechanism)	Generation (computational)
FUF, PENMAN	⊖ Semantic, few meanings	⊕ Linear	⊖ Deterministic	⊕ Linear
Nitrogen, HALogen	⊖ Syntactically/ Semantically grounded	⊖ Exponential	⊕ Non−deterministic via intersection with probabilistic LMs	⊕ Efficient Run−time Optimal Solution
Fergus, Amalgam	⊖ Syntactic dependencies	⊕ Linear	⊕ Non−deterministic via intersection with probabilistic LMs	⊕ Efficient Run−time Optimal Solution
IDL (Nederhof&Satta 2004)	⊕ Word/Phrase based	⊕ Linear	⊖ Deterministic via intersection with CFGs	⊖ Efficient Run−time All Solutions
IDL (this paper)	⊕ Word/Phrase based	⊕ Linear	⊕ Non−deterministic via intersection with probabilistic LMs	⊕ Efficient Run−time Optimal Solution

Table 1: Comparison of the present proposal with current NLG systems.

In Table 1, we present a summary of the representation and generation characteristics of current NLG systems. We mark by $\oplus$ characteristics that are needed/desirable in a generation component for text-to-text applications, and by $\ominus$ characteristics that make the proposal inapplicable or problematic. For instance, as already argued, the representation formalism of all previous proposals except for IDL is problematic ($\ominus$) for text-to-text applications. The IDL formalism, while applicable to text-to-text applications, has the additional desirable property that it is a compact representation, while formalisms such as word-lattices and non-recursive CFGs can have exponential size in the number of words available for generation (Nederhof and Satta, 2004).

While the IDL representational properties are all desirable, the generation mechanism proposed for IDL by Nederhof and Satta (2004) is problematic ($\ominus$), because it does not allow for scoring and ranking of candidate realizations. Their generation mechanism, while computationally efficient, involves intersection with context free grammars, and therefore works by excluding all realizations that are not accepted by a CFG and including (without ranking) all realizations that are accepted.

The approach to generation taken in this paper is presented in the last row in Table 1, and can be summarized as a $\oplus$ tiling of generation characteristics of previous proposals (see the shaded area in Table 1). Our goal is to provide an optimal generation framework for text-to-text applications, in which the representation formalism, the generation mechanism, and the computational properties are all needed and desirable ($\oplus$). Toward this goal, we present a new generation mechanism that intersects IDL-expressions with probabilistic language models. The generation mechanism implements new algorithms, which cover a wide spectrum of run-time behaviors (from linear to exponential), depending on the complexity of the input. We also present theoretical results concerning the correctness and the efficiency input IDL-expression) of our algorithms.

We evaluate these algorithms by performing experiments on a challenging word-ordering task. These experiments are carried out under a high-complexity generation scenario: find the most probable sentence realization under an n-gram language model for IDL-expressions encoding bags-of-words of size up to 25 (up to 10^{25} possible realizations!). Our evaluation shows that the proposed algorithms are able to cope well with such orders of complexity, while maintaining high levels of accuracy.

2 The IDL Language for NLG

2.1 IDL-expressions

IDL-expressions have been proposed by Nederhof & Satta (2004) (henceforth N&S) as a representation for finite languages, and are created from strings using four operators: concatenation ($\cdot$), interleave ($\|$), disjunction ($\vee$), and lock ($\times$). The semantics of IDL-expressions is given in terms of sets of strings.

The concatenation ($\cdot$) operator takes two arguments, and uses the strings encoded by its argument expressions to obtain concatenated strings that respect the order of the arguments; e.g., $\mathrm{a} \cdot \mathrm{b}$ encodes the singleton set $\{\mathrm{ab}\}$. The $\mathcal{I}$nterleave ($\|$) operator interleaves the strings encoded by its argument expressions; e.g., $\|(\mathrm{a} \cdot \mathrm{b}, \mathrm{c})$ encodes the set $\{\mathrm{cab}, \mathrm{acb}, \mathrm{abc}\}$. The $\mathcal{D}$isjunction ($\vee$) operator allows a choice among the strings encoded by its argument expressions; e.g., $\vee(\mathrm{a}, \mathrm{b})$ encodes the set $\{\mathrm{a}, \mathrm{b}\}$. The $\mathcal{L}$ock ($\times$) operator takes only one argument, and "locks-in" the strings encoded by its argument expression, such that no additional material can be interleaved; e.g., $\|(\times(\mathrm{a} \cdot \mathrm{b}), \mathrm{c})$ encodes the set $\{\mathrm{cab}, \mathrm{abc}\}$.

Consider the following IDL-expression:

$$\|(\mathit{finally}, \vee(\times(\mathit{the} \cdot \mathit{prisoners}), \times(\mathit{the} \cdot \mathit{captives})) \cdot \mathit{were} \cdot \mathit{released}) \qquad (1)$$

The concatenation ($\cdot$) operator captures precedence constraints, such as the fact that a determiner like *the* appears before the noun it determines. The lock ($\times$) operator enforces phrase-encoding constraints, such as the fact that *the captives* is a phrase which should be used as a whole. The disjunction ($\vee$) operator allows for multiple word/phrase choice (e.g., *the prisoners* versus *the captives*), and the interleave ($\|$) operator allows for word-order freedom, i.e., word order underspecification at meaning representation level. Among the strings encoded by IDL-expression 1 are the following:

finally the prisoners were released
the captives finally were released
the prisoners were finally released

The following strings, however, are not part of the language defined by IDL-expression 1:

the finally captives were released
the prisoners were released
finally the captives released were

The first string is disallowed because the $\times$ operator locks the phrase *the captives*. The second string is not allowed because the $\|$ operator requires all its arguments to be represented. The last string violates the order imposed by the precedence operator between *were* and *released*.

2.2 IDL-graphs

IDL-expressions are a convenient way to compactly represent finite languages. However, IDL-expressions do not directly allow formulations of algorithms to process them. For this purpose, an equivalent representation is introduced by N&S, called IDL-graphs. We refer the interested reader to the formal definition provided by N&S, and provide here only an intuitive description of IDL-graphs.

We illustrate in Figure 1 the IDL-graph corresponding to IDL-expression 1. In this graph, vertices v_s and v_e are called initial and final, respectively. Vertices v_0, v_2 with in-going $\vdash$-labeled edges, and v_1, v_{20} with out-going $\dashv$-labeled edges, for example, result from the expansion of the $\|$ operator, while vertices v_3, v_4 with in-going ϵ-labeled edges, and v_9, v_{14} with out-going ϵ-labeled edges result from the expansion of the $\vee$ operator. Vertices v_5 to v_8 and v_{10} to v_{13} result from the expansion of the two $\times$ operators, respectively. These latter vertices are also shown to have rank 1, as opposed to rank 0 (not shown) assigned to all other vertices. The ranking of vertices in an IDL-graph is needed to enforce a higher priority on the processing of the higher-ranked vertices, such that the desired semantics for the lock operator is preserved.

With each IDL-graph $G(\pi)$ we can associate a finite language: the set of strings that can be generated by an IDL-specific traversal of $G(\pi)$, starting from v_s and ending in v_e. An IDL-expression π and its corresponding IDL-graph $G(\pi)$ are said to be equivalent because they generate the same finite language, denoted $L(\pi)$.

2.3 IDL-graphs and Finite-State Acceptors

To make the connection with the formulation of our algorithms, in this section we link the IDL formalism with the more classical formalism of finite-state acceptors (FSA) (Hopcroft and Ullman, 1979). The FSA representation can naturally encode precedence and multiple choice, but it lacks primitives corresponding to the interleave ($\|$) and lock ($\times$) operators. As such, an FSA representation must explicitly enumerate all possible interleavings, which are implicitly captured in an IDL representation. This correspondence between implicit and explicit interleavings is naturally handled by the notion of a cut of an IDL-graph $G(\pi)$.

Intuitively, a cut through $G(\pi)$ is a set of vertices that can be reached *simultaneously* when traversing $G(\pi)$ from the initial node to the final node, following the branches as prescribed by the encoded $\mathcal{I}$, $\mathcal{D}$, and $\mathcal{L}$ operators, in an attempt to produce a string in $L(\pi)$. More precisely, the initial vertex v_s is considered a cut (Figure 2 (a)). For each vertex in a given cut, we create a new cut by replacing the start vertex of some edge with the end vertex of that edge, observing the following rules:

- the vertex that is the start of several edges labeled using the special symbol $\vdash$ is replaced by a sequence of all the end vertices of these edges (for example, v_0v_2 is a cut derived from v_s (Figure 2 (b))); a mirror rule handles the special symbol $\dashv$;

- the vertex that is the start of an edge labeled using vocabulary items or ϵ is replaced by the end vertex of that edge (for example, v_1v_2, v_0v_3, v_0v_5, v_0v_6 are cuts derived from v_0v_2, v_0v_2,

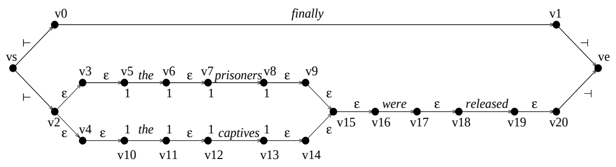


Figure 1: The IDL-graph corresponding to the IDL-expression $\|(\mathit{finally}, \vee(\times(\mathit{the} \cdot \mathit{prisoners}), \times(\mathit{the} \cdot \mathit{captives})) \cdot \mathit{were} \cdot \mathit{released})$.

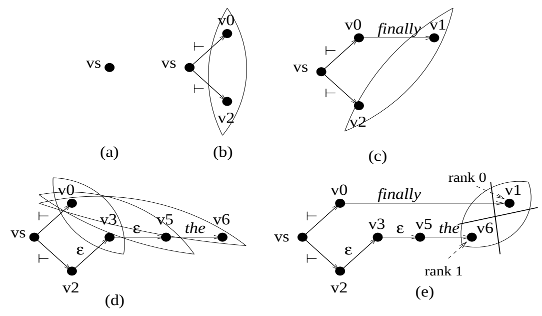


Figure 2: Cuts of the IDL-graph in Figure 1 (a-d). A non-cut is presented in (e).

v_0v_3, and v_0v_5, respectively, see Figure 2 (c-d)), only if the end vertex is not lower ranked than any of the vertices already present in the cut (for example, v_1v_6 is *not* a cut that can be derived from v_0v_6, see Figure 2 (e)).

Note the last part of the second rule, which restricts the set of cuts by using the ranking mechanism. If one would allow v_1v_6 to be a cut, one would imply that *finally* may appear inserted between the words of the locked phrase *the prisoners*.

We now link the IDL formalism with the FSA formalism by providing a mapping from an IDL-graph $G(\pi)$ to an acyclic finite-state acceptor $A(\pi)$. Because both formalisms are used for representing finite languages, they have equivalent representational power. The IDL representation is much more compact, however, as one can observe by comparing the IDL-graph in Figure 1 with the equivalent finite-state acceptor $A(\pi)$ in Figure 3. The set of states of $A(\pi)$ is the set of cuts of $G(\pi)$. The initial state of the finite-state acceptor is the state corresponding to cut v_s, and the final states of the finite-state acceptor are the state corresponding to cuts that contain v_e. In what follows, we denote a state of $A(\pi)$ by the name of the cut to which it corresponds. A transition labeled α in $A(\pi)$ between state $[v'_i \ldots v'_k \ldots v'_j]$ and state $[v''_i \ldots v''_k \ldots v''_j]$ occurs if there is an edge (v'_k, α, v''_k) in $G(\pi)$. For the example in Figure 3, the transition labeled *were* between states $[v_0v_{16}]$ and $[v_0v_{17}]$ occurs because of the edge labeled *were* between nodes v_{16} and v_{17} (Figure 1), whereas the transition labeled *finally* between states $[v_0v_{16}]$ and $[v_1v_{16}]$ occurs because of the edge labeled *finally* between nodes v_0 and v_1 (Figure 1). The two representations $G(\pi)$ and $A(\pi)$ are equivalent in the sense that the language generated by IDL-graph $G(\pi)$ is the same as the language accepted by FSA $A(\pi)$.

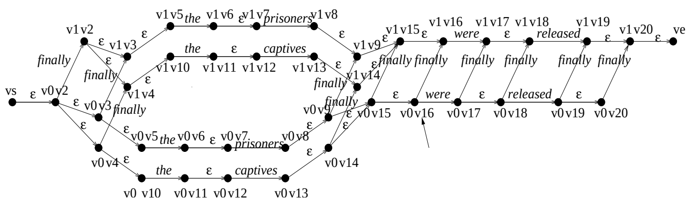


Figure 3: The finite-state acceptor corresponding to the IDL-graph in Figure 1.

It is not hard to see that the conversion from the IDL representation to the FSA representation destroys the compactness property of the IDL formalism, because of the explicit enumeration of all possible interleavings, which causes certain labels to appear repeatedly in transitions. For example, a transition labeled *finally* appears 11 times in the finite-state acceptor in Figure 3, whereas an edge labeled *finally* appears only once in the IDL-graph in Figure 1.

3 Computational Properties of IDL-expressions

3.1 IDL-graphs and Weighted Finite-State Acceptors

As mentioned in Section 1, the generation mechanism we propose performs an intersection of IDL-expressions with n-gram language models. Following (Mohri et al., 2002; Knight and Graehl, 1998), we implement language models using weighted finite-state acceptors (wFSA). In Section 2.3, we presented a mapping from an IDL-graph $G(\pi)$ to a finite-state acceptor $A(\pi)$. From such a finite-state acceptor $A(\pi)$, we arrive at a weighted finite-state acceptor $W(\pi)$, by splitting the states of $A(\pi)$ according to the information needed by the language model to assign weights to transitions. For example, under a bigram language model LM, state $[v_1 v_{16}]$ in Figure 3 must be split into three different states, $[prisoners, v_1 v_{16}]$, $[captives, v_1 v_{16}]$, and $[finally, v_1 v_{16}]$, according to which (non-epsilon) transition was last used to reach this state. The transitions leaving these states have the same labels as those leaving state $[v_1 v_{16}]$, and are now weighted using the language model probability distributions $p_{LM}(\cdot|prisoners)$, $p_{LM}(\cdot|captives)$, and $p_{LM}(\cdot|finally)$, respectively.

Note that, at this point, we already have a naïve algorithm for intersecting IDL-expressions with n-gram language models. From an IDL-expression π, following the mapping $\pi \rightarrow G(\pi) \rightarrow A(\pi) \rightarrow W(\pi)$, we arrive at a weighted finite-state acceptor, on which we can use a single-source shortest-path algorithm for directed acyclic graphs (Cormen et al., 2001) to extract the realization corresponding to the most probable path. The problem with this algorithm, however, is that the premature unfolding of the IDL-graph into a finite-state acceptor destroys the representation compactness of the IDL representation. For this reason, we devise algorithms that, although similar in spirit with the single-source shortest-path algorithm for directed acyclic graphs, perform on-the-fly unfolding of the IDL-graph, with a mechanism to control the unfolding based on the scores of the paths already unfolded. Such an approach has the advantage that prefixes that are extremely unlikely under the language model may be regarded as not so promising, and parts of the IDL-expression that contain them may not be unfolded, leading to significant savings.

3.2 Generation via Intersection of IDL-expressions with Language Models

Algorithm IDL-NGLM-BFS The first algorithm that we propose is algorithm IDL-NGLM-BFS in Figure 4. The algorithm builds a weighted finite-state acceptor W corresponding to an IDL-graph G incrementally, by keeping track of a set of active states, called $active$. The incrementality comes from creating new transitions and states in W originating in these active states, by unfolding the IDL-graph G; the set of newly unfolded states is called $unfold$. The new transitions in W are weighted ac-

```
IDL-NGLM-BFS$(G, LM)$
1  $active \leftarrow \{[vs^G]\}$
2  $flag \leftarrow 1$
3  while $flag$
4        do $unfold \leftarrow$ UNFOLDIDLG$(active, G)$
5           EVALUATENGLM$(unfold, LM)$
6           if FINALIDLG$(unfold, G)$
7              then $flag \leftarrow 0$
8           $active \leftarrow unfold$
9  return $active$
```

Figure 4: Pseudo-code for intersecting an IDL-graph G with an n-gram language model LM using incremental unfolding and breadth-first search.

cording to the language model. If a final state of W is not yet reached, the while loop is closed by making the $unfold$ set of states to be the next set of $active$ states. Note that this is actually a breadth-first search (BFS) with incremental unfolding. This algorithm still unfolds the IDL-graph completely, and therefore suffers from the same drawback as the naïve algorithm.

The interesting contribution of algorithm IDL-NGLM-BFS, however, is the incremental unfolding. If, instead of line 8 in Figure 4, we introduce mechanisms to control which $unfold$ states become part of the $active$ state set for the next unfolding iteration, we obtain a series of more effective algorithms.

Algorithm IDL-NGLM-A* We arrive at algorithm IDL-NGLM-A* by modifying line 8 in Figure 4, thus obtaining the algorithm in Figure 5. We use as control mechanism a priority queue, $astarQ$, in which the states from $unfold$ are PUSH-ed, sorted according to an admissible heuristic function (Russell and Norvig, 1995). In the next iteration, $active$ is a singleton set containing the state POP-ed out from the top of the priority queue.

Algorithm IDL-NGLM-BEAM We arrive at algorithm IDL-NGLM-BEAM by again modifying line 8 in Figure 4, thus obtaining the algorithm in Figure 6. We control the unfolding using a probabilistic beam $beam$, which, via the BEAMSTATES function, selects as $active$ states only the states in

```
IDL-NGLM-A*$(G, LM)$
1  $active \leftarrow \{[vs^G]\}$
2  $flag \leftarrow 1$
3  while $flag$
4      do $unfold \leftarrow$ UNFOLDIDLG$(active, G)$
5         EVALUATENGLM$(unfold, LM)$
6         if FINALIDLG$(unfold, G)$
7           then $flag \leftarrow 0$
8         for each $state$ in $unfold$
              do PUSH$(astarQ, state)$
          $active \leftarrow$ POP$(astarQ)$
9  return $active$
```

Figure 5: Pseudo-code for intersecting an IDL-graph G with an n-gram language model LM using incremental unfolding and A* search.

```
IDL-NGLM-BEAM$(G, LM, beam)$
1  $active \leftarrow \{[vs^G]\}$
2  $flag \leftarrow 1$
3  while $flag$
4      do $unfold \leftarrow$ UNFOLDIDLG$(active, G)$
5         EVALUATENGLM$(unfold, LM)$
6         if FINALIDLG$(unfold, G)$
7           then $flag \leftarrow 0$
8         $active \leftarrow$ BEAMSTATES$(unfold, beam)$
9  return $active$
```

Figure 6: Pseudo-code for intersecting an IDL-graph G with an n-gram language model LM using incremental unfolding and probabilistic beam search.

$unfold$ reachable with a probability higher or equal to the current maximum probability times the probability beam $beam$.

3.3 Computing Admissible Heuristics for IDL-expressions

The IDL representation is ideally suited for computing accurate admissible heuristics under language models. These heuristics are needed by the IDL-NGLM-A* algorithm, and are also employed for pruning by the IDL-NGLM-BEAM algorithm.

For each state S in a weighted finite-state acceptor W corresponding to an IDL-graph G, one can efficiently extract from G – without further unfolding – the set[1] of all edge labels that can be used to reach the final states of W. This set of labels, denoted FE^{all}_S, is an overestimation of the set of future events reachable from S, because the labels under the $\vee$ operators are all considered. From FE^{all}_S and the n-1 labels (when using an n-gram language model) recorded in state S we obtain the set of label sequences of length n-1. This set, denoted FCE_S, is an (over)estimated set of possible future conditioning events for state S, guaranteed to contain the most cost-efficient future conditioning events for state S. Using FCE_S, one needs to extract from FE^{all}_S the set of most cost-efficient future events from under each $\vee$ operator. We use this set, denoted FE_S, to arrive at an admissible heuristic for state S under a language model LM, using Equation 2:

$$h(S) = \Sigma_{e \in FE_S} - \log(\max_{ce \in FCE_S} p_{LM}(e|ce)) \quad (2)$$

If $h^*(S)$ is the true future cost for state S, we guarantee that $h(S) \leq h^*(S)$ from the way FE_S and FCE_S are constructed. Note that, as it usually happens with admissible heuristics, we can make $h(S)$ come arbitrarily close to $h^*(S)$, by computing increasingly better approximations FCE_S of FCE^*_S. Such approximations, however, require increasingly advanced unfoldings of the IDL-graph G (a complete unfolding of G for state S gives $FCE_S = FCE^*_S$, and consequently $h(S) = h^*(S)$). It follows that arbitrarily accurate admissible heuristics exist for IDL-expressions, but computing them on-the-fly requires finding a balance between the time and space requirements for computing better heuristics and the speed-up obtained by using them in the search algorithms.

3.4 Formal Properties of IDL-NGLM algorithms

The following theorem states the correctness of our algorithms, in the sense that they find the maximum probability path encoded by an IDL-graph under an n-gram language model.

Theorem 1 *Let π be an IDL-expression, G(π) its IDL-graph, and W(π) its wFSA under an n-gram language model LM. Algorithms* IDL-NGLM-BFS *and* IDL-NGLM-A* *find the*

[1]Actually, these are multisets, as we treat multiply-occurring labels as separate items.

path of maximum probability under LM. Algorithm IDL-NGLM-BEAM *finds the path of maximum probability under LM, if all states in W(π) along this path are selected by its* BEAMSTATES *function.*

The proof of the theorem follows directly from the correctness of the BFS and A* search, and from the condition imposed on the beam search.

The next theorem characterizes the run-time complexity of these algorithms, in terms of an input IDL-expression π and its corresponding IDL-graph $G(\pi)$ complexity. There are three factors that linearly influence the run-time complexity of our algorithms: a is the maximum number of nodes in $G(\pi)$ needed to represent a state in $A(\pi)$ – a depends solely on π; w is the maximum number of nodes in $G(\pi)$ needed to represent a state in $W(\pi)$ – w depends on π and n, the length of the context used by the n-gram language model; and K is the number of states of $W(\pi)$ – K also depends on π and n. Of these three factors, K is by far the predominant one, and we simply call K the complexity of an IDL-expression.

Theorem 2 *Let π be an IDL-expression, $G(\pi)$ its IDL-graph, $A(\pi)$ its FSA, and $W(\pi)$ its wFSA under an n-gram language model. Let $V[A(\pi)]$ be the set of states of $A(\pi)$, and $V[W(\pi)]$ the set of states of $W(\pi)$. Let also $a = \max_{c\in V[A(\pi)]}|c|$, $w = \max_{c\in V[W(\pi)]}|c|$, and $K = |V[W(\pi)]|$. Algorithms* IDL-NGLM-BFS *and* IDL-NGLM-BEAM *have run-time complexity* $\mathrm{O}(awK)$. *Algorithm* IDL-NGLM-A* *has run-time complexity* $\mathrm{O}(awK\log K)$.

We omit the proof here due to space constraints. The fact that the run-time behavior of our algorithms is linear in the complexity of the input IDL-expression (with an additional log factor in the case of A* search due to priority queue management) allows us to say that our algorithms are efficient with respect to the task they accomplish.

We note here, however, that depending on the input IDL-expression, the task addressed can vary in complexity from linear to exponential. That is, for the intersection of an IDL-expression $\pi = \|(w_1,\ldots,w_n)$ (bag of n words) with a trigram language model, we have $a(\pi) = n$, $w(\pi) = n+2$, $K = c^n, c > 1$, and therefore a $\mathrm{O}(n^2c^n)$ complexity. This exponential complexity comes as no surprise given that the problem of intersecting an n-gram language model with a bag of words is known to be NP-complete (Knight, 1999). On the other hand, for intersecting an IDL-expression $\pi = w_1 \cdot \ldots \cdot w_n$ (sequence of n words) with a trigram language model, we have $a(\pi) = 1$, $w(\pi) = 3$, and $K = n$, and therefore an $\mathrm{O}(n)$ generation algorithm.

In general, for IDL-expressions for which a is bounded, which we expect to be the case for most practical problems, *our algorithms perform in polynomial time in the number of words available for generation.*

4 Evaluation of IDL-NGLM Algorithms

In this section, we present results concerning the performance of our algorithms on a word-ordering task. This task can be easily defined as follows: from a bag of words originating from some sentence, reconstruct the original sentence as faithfully as possible. In our case, from an original sentence such as *"the gifts are donated by american companies"*, we create the IDL-expression $\langle s\rangle \cdot \|(the, gifts, donated, companies, by, are, american) \cdot \langle /s\rangle$, from which some algorithm realizes a sentence such as *"donated by the american companies are gifts"*. Note the natural way we represent in an IDL-expression beginning and end of sentence constraints, using the $\cdot$ operator. Since this is generation from bag-of-words, the task is known to be at the high-complexity extreme of the run-time behavior of our algorithms. As such, we consider it a good test for the ability of our algorithms to scale up to increasingly complex inputs.

We use a state-of-the-art, publicly available toolkit[2] to train a trigram language model using Kneser-Ney smoothing, on 10 million sentences (170 million words) from the Wall Street Journal (WSJ), lower case and no final punctuation. The test data is also lower case (such that upper-case words cannot be hypothesized as first words), with final punctuation removed (such that periods cannot be hypothesized as final words), and consists of 2000 unseen WSJ sentences of length 3-7, and 2000 unseen WSJ sentences of length 10-25.

The algorithms we tested in this experiments were the ones presented in Section 3.2, plus two baseline algorithms. The first baseline algorithm, L, uses an

[2]http://www.speech.sri.com/projects/srilm/

inverse-lexicographic order for the bag items as its output, in order to get the word *the* on sentence initial position. The second baseline algorithm, G, is a greedy algorithm that realizes sentences by maximizing the probability of joining any two word sequences until only one sequence is left.

For the A* algorithm, an admissible cost is computed for each state S in a weighted finite-state automaton, as the sum (over all unused words) of the minimum language model cost (i.e., maximum probability) of each unused word when conditioning over all sequences of two words available at that particular state for future conditioning (see Equation 2, with $FE_S = FE_S^{all}$). These estimates are also used by the beam algorithm for deciding which IDL-graph nodes are not unfolded. We also test a greedy version of the A* algorithm, denoted A$^*_k$, which considers for unfolding only the nodes extracted from the priority queue which already unfolded a path of length greater than or equal to the maximum length already unfolded minus k (in this notation, the A* algorithm would be denoted A$^*_\infty$). For the beam algorithms, we use the notation B$_p$ to specify a probabilistic beam of size p, i.e., an algorithm that beams out the states reachable with probability less than the current maximum probability times p.

Our first batch of experiments concerns bags-of-words of size 3-7, for which exhaustive search is possible. In Table 2, we present the results on the word-ordering task achieved by various algorithms. We evaluate accuracy performance using two automatic metrics: an identity metric, ID, which measures the percent of sentences recreated exactly, and BLEU (Papineni et al., 2002), which gives the geometric average of the number of uni-, bi-, tri-, and four-grams recreated exactly. We evaluate the search performance by the percent of Search Errors made by our algorithms, as well as a percent figure of Estimated Search Errors, computed as the percent of searches that result in a string with a lower probability than the probability of the original sentence. To measure the impact of using IDL-expressions for this task, we also measure the percent of unfolding of an IDL graph with respect to a full unfolding. We report speed results as the average number of seconds per bag-of-words, when using a 3.0GHz CPU machine under a Linux OS.

The first notable result in Table 2 is the savings

ALG	ID (%)	BLEU	Search Errors (%)	Unfold (%)	Speed (sec./bag)
L	2.5	9.5	97.2 (95.8)	N/A	.000
G	30.9	51.0	67.5 (57.6)	N/A	.000
BFS	67.1	79.2	0.0 (0.0)	100.0	.072
A*	67.1	79.2	0.0 (0.0)	12.0	.010
A$^*_1$	60.5	74.8	21.1 (11.9)	3.2	.004
A$^*_2$	64.3	77.2	8.5 (4.0)	5.3	.005
B$_{0.2}$	65.0	78.0	9.2 (5.0)	7.2	.006
B$_{0.1}$	66.6	78.8	3.2 (1.7)	13.2	.011

Table 2: Bags-of-words of size 3-7: accuracy (ID, BLEU), Search Errors (and Estimated Search Errors), space savings (Unfold), and speed results.

achieved by the A* algorithm under the IDL representation. At no cost in accuracy, it unfolds only 12% of the edges, and achieves a 7 times speed-up, compared to the BFS algorithm. The savings achieved by not unfolding are especially important, since the exponential complexity of the problem is hidden by the IDL representation via the folding mechanism of the $\|$ operator. The algorithms that find sub-optimal solutions also perform well. While maintaining high accuracy, the A$^*_2$ and B$_{0.2}$ algorithms unfold only about 5-7% of the edges, at 12-14 times speed-up.

Our second batch of experiments concerns bag-of-words of size 10-25, for which exhaustive search is no longer possible (Table 3). Not only exhaustive search, but also full A* search is too expensive in terms of memory (we were limited to 2GiB of RAM for our experiments) and speed. Only the greedy versions A$^*_1$ and A$^*_2$, and the beam search using tight probability beams (0.2-0.1) scale up to these bag sizes. Because we no longer have access to the string of maximum probability, we report only the percent of Estimated Search Errors. Note that, in terms of accuracy, we get around 20% Estimated Search Errors for the best performing algorithms (A$^*_2$ and B$_{0.1}$), which means that 80% of the time the algorithms are able to find sentences of equal or better probability than the original sentences.

5 Conclusions

In this paper, we advocate that IDL expressions can provide an adequate framework for develop-

ALG	**ID** (%)	**BLEU**	**Est. Search Errors (%)**	**Speed** (sec./bag)
L	0.0	1.4	99.9	0.0
G	1.2	31.6	83.6	0.0
A_1^*	5.8	47.7	34.0	0.7
A_2^*	7.4	51.2	21.4	9.5
$B_{0.2}$	9.0	52.1	23.3	7.1
$B_{0.1}$	12.2	52.6	19.9	36.7

Table 3: Bags-of-words of size 10-25: accuracy (ID, BLEU), Estimated Search Errors, and speed results.

ing text-to-text generation capabilities. Our contribution concerns a new generation mechanism that implements intersection between an IDL expression and a probabilistic language model. The IDL formalism is ideally suited for our approach, due to its efficient representation and, as we show in this paper, efficient algorithms for intersecting, scoring, and ranking sentence realizations using probabilistic language models.

We present theoretical results concerning the correctness and efficiency of the proposed algorithms, and also present empirical results that show that our algorithms scale up to handling IDL-expressions of high complexity. Real-world text-to-text generation tasks, such as headline generation and machine translation, are likely to be handled graciously in this framework, as the complexity of IDL-expressions for these tasks tends to be lower than the complexity of the IDL-expressions we worked with in our experiments.

Acknowledgment

This work was supported by DARPA-ITO grant NN66001-00-1-9814.

References

Srinivas Bangalore and Owen Rambow. 2000. Using TAG, a tree model, and a language model for generation. In *Proceedings of the 1st International Natural Language Generation Conference*.

Regina Barzilay. 2003. *Information Fusion for Multi-document Summarization: Paraphrasing and Generation*. Ph.D. thesis, Columbia University.

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.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms*. The MIT Press and McGraw-Hill. Second Edition.

Simon Corston-Oliver, Michael Gamon, Eric K. Ringger, and Robert Moore. 2002. An overview of Amalgam: A machine-learned generation module. In *Proceedings of the International Natural Language Generation Conference*.

Michael Elhadad. 1991. FUF User manual — version 5.0. Technical Report CUCS-038-91, Department of Computer Science, Columbia University.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to automata theory, languages, and computation*. Addison-Wesley.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Kevin Knight and Vasileios Hatzivassiloglou. 1995. Two level, many-path generation. In *Proceedings of the Association of Computational Linguistics*.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Irene Langkilde-Geary. 2002. *A foundation for general-purpose natural language generation: sentence realization using probabilistic models of language*. Ph.D. thesis, University of Southern California.

Christian Matthiessen and John Bateman. 1991. *Text Generation and Systemic-Functional Linguistic*. Pinter Publishers, London.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88.

Mark-Jan Nederhof and Giorgio Satta. 2004. IDL-expressions: a formalism for representing and parsing finite languages in natural language processing. *Journal of Artificial Intelligence Research*, 21:287–317.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics (ACL-2002)*, pages 311–318, Philadelphia, PA, July 7-12.

Stuart Russell and Peter Norvig. 1995. *Artificial Intelligence. A Modern Approach*. Prentice Hall, Englewood Cliffs, New Jersey.

Probabilistic CFG with latent annotations

Takuya Matsuzaki† Yusuke Miyao† Jun'ichi Tsujii†‡
†Graduate School of Information Science and Technology, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033
‡CREST, JST(Japan Science and Technology Agency)
Honcho 4-1-8, Kawaguchi-shi, Saitama 332-0012
{matuzaki, yusuke, tsujii}@is.s.u-tokyo.ac.jp

Abstract

This paper defines a generative probabilistic model of parse trees, which we call PCFG-LA. This model is an extension of PCFG in which non-terminal symbols are augmented with latent variables. Fine-grained CFG rules are automatically induced from a parsed corpus by training a PCFG-LA model using an EM-algorithm. Because exact parsing with a PCFG-LA is NP-hard, several approximations are described and empirically compared. In experiments using the Penn WSJ corpus, our automatically trained model gave a performance of 86.6% (F_1, sentences ≤ 40 words), which is comparable to that of an unlexicalized PCFG parser created using extensive manual feature selection.

1 Introduction

Variants of PCFGs form the basis of several broad-coverage and high-precision parsers (Collins, 1999; Charniak, 1999; Klein and Manning, 2003). In those parsers, the strong conditional independence assumption made in vanilla treebank PCFGs is weakened by annotating non-terminal symbols with many 'features' (Goodman, 1997; Johnson, 1998). Examples of such features are head words of constituents, labels of ancestor and sibling nodes, and subcategorization frames of lexical heads. Effective features and their good combinations are normally explored using trial-and-error.

This paper defines a generative model of parse trees that we call PCFG with latent annotations (PCFG-LA). This model is an extension of PCFG models in which non-terminal symbols are annotated with latent variables. The latent variables work just like the features attached to non-terminal symbols. A fine-grained PCFG is automatically induced from parsed corpora by training a PCFG-LA model using an EM-algorithm, which replaces the manual feature selection used in previous research.

The main focus of this paper is to examine the effectiveness of the automatically trained models in parsing. Because exact inference with a PCFG-LA, i.e., selection of the most probable parse, is NP-hard, we are forced to use some approximation of it. We empirically compared three different approximation methods. One of the three methods gives a performance of 86.6% (F_1, sentences ≤ 40 words) on the standard test set of the Penn WSJ corpus.

Utsuro et al. (1996) proposed a method that automatically selects a proper level of generalization of non-terminal symbols of a PCFG, but they did not report the results of parsing with the obtained PCFG. Henderson's parsing model (Henderson, 2003) has a similar motivation as ours in that a derivation history of a parse tree is compactly represented by induced hidden variables (hidden layer activation of a neural network), although the details of his approach is quite different from ours.

2 Probabilistic model

PCFG-LA is a generative probabilistic model of parse trees. In this model, an observed parse tree is considered as an incomplete data, and the corre-

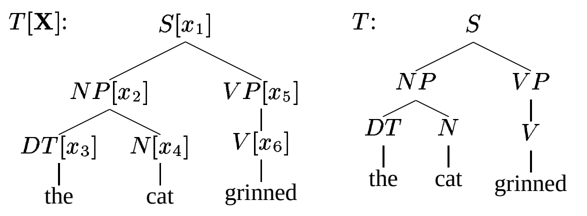


Figure 1: Tree with latent annotations $T[\mathbf{X}]$ (complete data) and observed tree T (incomplete data).

sponding complete data is a tree with *latent annotations*. Each non-terminal node in the complete data is labeled with a complete symbol of the form $A[x]$, where A is the non-terminal symbol of the corresponding node in the observed tree and x is a *latent annotation symbol*, which is an element of a fixed set H.

A complete/incomplete tree pair of the sentence, "*the cat grinned*," is shown in Figure 2. The complete parse tree, $T[\mathbf{X}]$ (left), is generated through a process just like the one in ordinary PCFGs, but the non-terminal symbols in the CFG rules are annotated with latent symbols, $\mathbf{X} = (x_1, x_2, \dots)$. Thus, the probability of the complete tree ($T[\mathbf{X}]$) is

$$\begin{aligned}
&P(T[\mathbf{X}]) \\
&= \pi(S[x_1]) \times \beta(S[x_1] \to NP[x_2]VP[x_5]) \\
&\times \beta(NP[x_2] \to DT[x_3]N[x_4]) \\
&\times \beta(DT[x_3] \to \textit{the}) \times \beta(N[x_4] \to \textit{cat}) \\
&\times \beta(VP[x_5] \to V[x_6]) \times \beta(V[x_6] \to \textit{grinned}),
\end{aligned}$$

where $\pi(S[x_1])$ denotes the probability of an occurrence of the symbol $S[x_1]$ at a root node and $\beta(r)$ denotes the probability of a CFG rule r. The probability of the observed tree $P(T)$ is obtained by summing $P(T[\mathbf{X}])$ for all the assignments to latent annotation symbols, $\mathbf{X}$:

$$P(T) = \sum_{x_1 \in H} \sum_{x_2 \in H} \cdots \sum_{x_6 \in H} P(T[\mathbf{X}]). \qquad (1)$$

Using dynamic programming, the theoretical bound of the time complexity of the summation in Eq. 1 is reduced to be proportional to the number of non-terminal nodes in a parse tree. However, the calculation at node n still has a cost that exponentially grows with the number of n's daughters because we must sum up the probabilities of $|H|^{d+1}$ combinations of latent annotation symbols for a node with d daughters. We thus took a kind of transformation/detransformation approach, in which a tree is binarized before parameter estimation and restored to its original form after parsing. The details of the binarization are explained in Section 4.

Using syntactically annotated corpora as training data, we can estimate the parameters of a PCFG-LA model using an EM algorithm. The algorithm is a special variant of the inside-outside algorithm of Pereira and Schabes (1992). Several recent work also use similar estimation algorithm as ours, i.e, inside-outside re-estimation on parse trees (Chiang and Bikel, 2002; Shen, 2004).

The rest of this section precisely defines PCFG-LA models and briefly explains the estimation algorithm. The derivation of the estimation algorithm is largely omitted; see Pereira and Schabes (1992) for details.

2.1 Model definition

We define a PCFG-LA $\mathcal{M}$ as a tuple $\mathcal{M} = \langle N_{\text{nt}}, N_{\text{t}}, H, R, \pi, \beta \rangle$, where

$$\begin{aligned}
N_{\text{nt}}&: \text{a set of observable non-terminal symbols} \\
N_{\text{t}}&: \text{a set of terminal symbols} \\
H&: \text{a set of latent annotation symbols} \\
R&: \text{a set of observable CFG rules} \\
\pi(A[x])&: \text{the probability of the occurrence} \\
&\quad \text{of a complete symbol } A[x] \text{ at a root node} \\
\beta(r)&: \text{the probability of a rule } r \in R[H].
\end{aligned}$$

We use $A, B, \dots$ for non-terminal symbols in N_{nt}; $w_1, w_2, \dots$ for terminal symbols in N_{t}; and $x, y, \dots$ for latent annotation symbols in H. $N_{\text{nt}}[H]$ denotes the set of complete non-terminal symbols, i.e., $N_{\text{nt}}[H] = \{A[x] \mid A \in N_{\text{nt}}, x \in H\}$. Note that latent annotation symbols are not attached to terminal symbols.

In the above definition, R is a set of CFG rules of observable (i.e., not annotated) symbols. For simplicity of discussion, we assume that R is a CNF grammar, but extending to the general case is straightforward. $R[H]$ is the set of CFG rules of complete symbols, such as $V[x] \to$ *grinned* or $S[x] \to NP[y]VP[z]$. More precisely,

$$\begin{aligned}
R[H] = &\{(A[x] \to w) \mid (A \to w) \in R; x \in H\} \cup \\
&\{(A[x] \to B[y]C[z]) \mid (A \to BC) \in R; x, y, z \in H\}.
\end{aligned}$$

We assume that non-terminal nodes in a parse tree T are indexed by integers $i = 1, \ldots, m$, starting from the root node. A complete tree is denoted by $T[\mathbf{X}]$, where $\mathbf{X} = (x_1, \ldots, x_m) \in H^m$ is a vector of latent annotation symbols and x_i is the latent annotation symbol attached to the i-th non-terminal node.

We do not assume any structured parametrizations in β and π; that is, each $\beta(r)$ $(r \in R[H])$ and $\pi(A[x])$ $(A[x] \in N_{\text{nt}}[H])$ is itself a parameter to be tuned. Therefore, an annotation symbol, say, x, generally does not express any commonalities among the complete non-terminals annotated by x, such as $A[x], B[x], etc.$

The probability of a complete parse tree $T[\mathbf{X}]$ is defined as

$$P(T[\mathbf{X}]) = \pi(A_1[x_1]) \prod_{r \in D_{T[\mathbf{X}]}} \beta(r), \qquad (2)$$

where $A_1[x_1]$ is the label of the root node of $T[\mathbf{X}]$ and $D_{T[\mathbf{X}]}$ denotes the multiset of annotated CFG rules used in the generation of $T[\mathbf{X}]$. We have the probability of an observable tree T by marginalizing out the latent annotation symbols in $T[\mathbf{X}]$:

$$P(T) = \sum_{X \in H^m} \pi(A_1[x_1]) \prod_{r \in D_{T[\mathbf{X}]}} \beta(r), \qquad (3)$$

where m is the number of non-terminal nodes in T.

2.2 Forward-backward probability

The sum in Eq. 3 can be calculated using a dynamic programming algorithm analogous to the forward algorithm for HMMs. For a sentence $w_1 w_2 \ldots w_n$ and its parse tree T, backward probabilities $b_T^i(x)$ are recursively computed for the i-th non-terminal node and for each $x \in H$. In the definition below, $N_i \in N_{\text{nt}}$ denotes the non-terminal label of the i-th node.

- If node i is a pre-terminal node above a terminal symbol w_j, then $b_T^i(x) = \beta(N_i[x] \rightarrow w_j)$.
- Otherwise, let j and k be the two daughter nodes of i. Then

$$b_T^i(x) = \sum_{x_j, x_k \in H} \begin{array}{c} \beta(N_i[x] \rightarrow N_j[x_j] N_k[x_k]) \\ \times\, b_T^j(x_j)\, b_T^k(x_k). \end{array}$$

Using backward probabilities, $P(T)$ is calculated as $P(T) = \sum_{x_1 \in H} \pi(N_1[x_1])\, b_T^1(x_1)$.

We define forward probabilities $f_T^i(x)$, which are used in the estimation described below, as follows:

- If node i is the root node (i.e., i = 1), then $f_T^i(x) = \pi(N_i[x])$.
- If node i has a right sibling k, let j be the mother node of i. Then

$$f_T^i(x) = \sum_{x_j, x_k \in H} \begin{array}{c} \beta(N_j[x_j] \rightarrow N_i[x] N_k[x_k]) \\ \times\, f_T^j(x_j)\, b_T^k(x_k). \end{array}$$

- If node i has a left sibling, $f_T^i(x)$ is defined analogously.

2.3 Estimation

We now derive the EM algorithm for PCFG-LA, which estimates the parameters $\theta = (\beta, \pi)$. Let $\mathbf{T} = \{T_1, T_2, \ldots\}$ be the training set of parse trees and $N_1^i, \ldots, N_{m_i}^i$ be the labels of non-terminal nodes in T_i. Like the derivations of the EM algorithms for other latent variable models, the update formulas for the parameters, which update the parameters from θ to $\theta' = (\beta', \pi')$, are obtained by constrained optimization of $Q(\theta'|\theta)$, which is defined as

$$Q(\theta'|\theta) = \sum_{T_i \in \mathbf{T}} \sum_{\mathbf{X}_i \in H^{m_i}} P_\theta(\mathbf{X}_i | T_i) \log P_{\theta'}(T_i[\mathbf{X}_i]),$$

where P_θ and $P_{\theta'}$ denote probabilities under θ and θ', and $P(\mathbf{X}|T)$ is the conditional probability of latent annotation symbols given an observed tree T, i.e., $P(\mathbf{X}|T) = P(T[\mathbf{X}])/P(T)$. Using the Lagrange multiplier method and re-arranging the results using the backward and forward probabilities, we obtain the update formulas in Figure 2.

3 Parsing with PCFG-LA

In theory, we can use PCFG-LAs to parse a given sentence w by selecting the most probable parse:

$$T_{best} = \operatorname*{argmax}_{T \in G(w)} P(T|w) = \operatorname*{argmax}_{T \in G(w)} P(T), \qquad (4)$$

where $G(w)$ denotes the set of possible parses for w under the observable grammar R. While the optimization problem in Eq. 4 can be efficiently solved

$$\beta'(A[x] \to B[y]C[z]) = Z_{A[x]}^{-1} \sum_{T_i \in \mathbf{T}} P(T_i)^{-1} \times$$
$$\sum_{(j,k,l) \in \text{Covered}(T_i, A \to BC)} f_{T_i}^{j}(x)\, \beta(A[x] \to B[y]C[z])\, b_{T_i}^{k}(y)\, b_{T_i}^{l}(z)$$
$$\beta'(A[x] \to w) = Z_{A[x]}^{-1} \sum_{T_i \in \mathbf{T}} P(T_i)^{-1} \sum_{j \in \text{Covered}(T_i, A \to w)} f_{T_i}^{j}(x)\, \beta(A[x] \to w)$$
$$\pi'(A[x]) = |\mathbf{T}|^{-1} \sum_{T_i \in \text{Root}(\mathbf{T}, A)} P(T_i)^{-1} \pi(A[x])\, b_{T_i}^{1}(x)$$
$$Z_{A[x]} = \sum_{T_i \in \mathbf{T}} P(T_i)^{-1} \sum_{j \in \text{Labeled}(T_i, A)} f_{T_i}^{j}(x)\, b_{T_i}^{j}(x)$$
$$\text{Covered}(T_i, A \to BC) =$$
$$\{(j,k,l) \mid N_j^i \to N_k^i N_l^i \in D_{T_i}; (N_j^i, N_k^i, N_l^i) = (A, B, C)\}$$
$$\text{Covered}(T_i, A \to w) = \{j \mid N_j^i \to w \in D_{T_i}; N_j^i = A\}$$
$$\text{Labeled}(T_i, A) = \{j \mid N_j^i = A\}$$
$$\text{Root}(\mathbf{T}, A) = \{T_i \in \mathbf{T} \mid \text{the root of } T_i \text{ is labeled with } A\}$$

Figure 2: Parameter update formulas.

for PCFGs using dynamic programming algorithms, the sum-of-products form of $P(T)$ in PCFG-LA models (see Eq. 2 and Eq. 3) makes it difficult to apply such techniques to solve Eq. 4.

Actually, the optimization problem in Eq. 4 is NP-hard for general PCFG-LA models. Although we omit the details, we can prove the NP-hardness by observing that a stochastic tree substitution grammar (STSG) can be represented by a PCFG-LA model in a similar way to one described by Goodman (1996a), and then using the NP-hardness of STSG parsing (Simaán, 2002).

The difficulty of the exact optimization in Eq. 4 forces us to use some approximations of it. The rest of this section describes three different approximations, which are empirically compared in the next section. The first method simply limits the number of candidate parse trees compared in Eq. 4; we first create N-best parses using a PCFG and then, within the N-best parses, select the one with the highest probability in terms of the PCFG-LA. The other two methods are a little more complicated, and we explain them in separate subsections.

3.1 Approximation by Viterbi complete trees

The second approximation method selects the best *complete* tree $T'[\mathbf{X}']$, that is,

$$T'[\mathbf{X}'] = \operatorname*{argmax}_{T \in G(w), \mathbf{X} \in H^{|\mathbf{X}|}} P(T[\mathbf{X}]). \quad (5)$$

We call $T'[\mathbf{X}']$ a Viterbi complete tree. Such a tree can be obtained in $O(|w|^3)$ time by regarding the PCFG-LA as a PCFG with annotated symbols.[1]

The observable part of the Viterbi complete tree $T'[\mathbf{X}']$ (i.e., T') does not necessarily coincide with the best observable tree T_{best} in Eq. 4. However, if T_{best} has some 'dominant' assignment $\mathbf{Y}$ to its latent annotation symbols such that $P(T_{best}[\mathbf{Y}]) \approx P(T_{best})$, then $P(T') \approx P(T_{best})$ because $P(T_{best}[\mathbf{Y}]) \le P(T'[\mathbf{X}'])$ and $P(T'[\mathbf{X}']) \le P(T')$, and thus T' and T_{best} are almost equally 'good' in terms of their marginal probabilities.

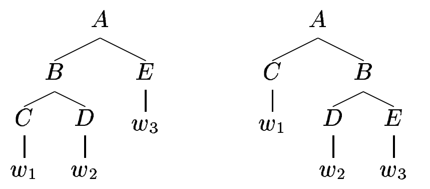
3.2 Viterbi parse in approximate distribution

In the third method, we approximate the true distribution $P(T|w)$ by a cruder distribution $Q(T|w)$, and then find the tree with the highest $Q(T|w)$ in polynomial time. We first create a packed representation of $G(w)$ for a given sentence w.[2] Then, the approximate distribution $Q(T|w)$ is created using the packed forest, and the parameters in $Q(T|w)$ are adjusted so that $Q(T|w)$ approximates $P(T|w)$ as closely as possible. The form of $Q(T|w)$ is that of a product of the parameters, just like the form of a PCFG model, and it enables us to use a Viterbi algorithm to select the tree with the highest $Q(T|w)$.

A packed forest is defined as a tuple $\langle I, \delta \rangle$. The first component, I, is a multiset of chart items of the form (A, b, e). A chart item $(A, b, e) \in I$ indicates that there exists a parse tree in $G(w)$ that contains a constituent with the non-terminal label A that spans

[1]For efficiency, we did not actually parse sentences with $R[H]$ but selected a Viterbi complete tree from a packed representation of candidate parses in the experiments in Section 4.

[2]In practice, fully constructing a packed representation of $G(w)$ has an unrealistically high cost for most input sentences. Alternatively, we can use a packed representation of a subset of $G(w)$, which can be obtained by parsing with beam thresholding, for instance. An approximate distribution $Q(T|w)$ on such subsets can be derived in almost the same way as one for the full $G(w)$, but the conditional distribution, $P(T|w)$, is renormalized so that the total mass for the subset sums to 1.



$$I = \{i_1, i_2, i_3, i_4, i_5, i_6\}$$
$$i_1 = (A,1,3), i_2 = (B,1,2), i_3 = (B,2,3),$$
$$i_4 = (C,1,1), i_5 = (D,2,2), i_6 = (E,3,3)$$
$$\delta(i_1) = \{(i_2,i_6),(i_3,i_4)\}$$
$$\delta(i_2) = \{(i_4,i_5)\}, \quad \delta(i_3) = \{(i_5,i_6)\}$$
$$\delta(i_4) = \{w_1\}, \delta(i_5) = \{w_2\}, \delta(i_6) = \{w_3\}$$

Figure 3: Two parse trees and packed representation of them.

from the b-th to e-th word in w. The second component, δ, is a function on I that represents dominance relations among the chart items in I; $\delta(i)$ is a set of possible daughters of i if i is not a pre-terminal node, and $\delta(i) = \{w_k\}$ if i is a pre-terminal node above w_k. Two parse trees for a sentence $w = w_1w_2w_3$ and a packed representation of them are shown in Figure 3.

We require that each tree $T \in G(w)$ has a unique representation as a set of connected chart items in I. A packed representation satisfying the uniqueness condition is created using the CKY algorithm with the observable grammar R, for instance.

The approximate distribution, $Q(T|w)$, is defined as a PCFG, whose CFG rules R_w is defined as $R_w = \{(i \to \eta) \mid i \in I; \eta \in \delta(i)\}$. We use $q(r)$ to denote the rule probability of rule $r \in R_w$ and $q_r(i)$ to denote the probability with which $i \in I$ is generated as a root node. We define $Q(T|w)$ as

$$Q(T|w) = q_r(i_1)\prod_{k=1}^{m} q(i_k \to \eta_k),$$

where the set of connected items $\{i_1, \ldots, i_m\} \subset I$ is the unique representation of T.

To measure the closeness of approximation by $Q(T|w)$, we use the 'inclusive' KL-divergence, $KL(P||Q)$ (Frey et al., 2000):

$$KL(P||Q) = \sum_{T \in G(w)} P(T|w) \log \frac{P(T|w)}{Q(T|w)}.$$

Minimizing $KL(P||Q)$ under the normalization constraints on q_r and q yields closed form solutions for q_r and q, as shown in Figure 4.

P_{in} and P_{out} in Figure 4 are similar to ordinary inside/outside probabilities. We define P_{in} as follows:

- If $i = (A,k,k) \in I$ is a pre-terminal node above w_k, then $P_{\text{in}}(i[x]) = \beta(A[x] \to w_k)$.
- Otherwise,

$$P_{\text{in}}(i[x]) = \sum_{jk \in \delta(i)} \sum_{y,z \in H} \begin{array}{l}\beta(A[x] \to B_j[y]C_k[z]) \\ \times P_{\text{in}}(j[y])P_{\text{in}}(k[z]),\end{array}$$

  where B_j and C_k denote non-terminal symbols of chart items j and k.

The outside probability, P_{out}, is calculated using P_{in} and PCFG-LA parameters along the packed structure, like the outside probabilities for PCFGs.

Once we have computed $q(i \to \eta)$ and $q_r(i)$, the parse tree T that maximizes $Q(T|w)$ is found using a Viterbi algorithm, as in PCFG parsing.

Several parsing algorithms that also use inside-outside calculation on packed chart have been proposed (Goodman, 1996b; Simaán, 2003; Clark and Curran, 2004). Those algorithms optimize some evaluation metric of parse trees other than the posterior probability $P(T|w)$, e.g., (expected) labeled constituent recall or (expected) recall rate of dependency relations contained in a parse. It is in contrast with our approach where (approximated) posterior probability is optimized.

4 Experiments

We conducted four sets of experiments. In the first set of experiments, the degree of dependency of trained models on initialization was examined because EM-style algorithms yield different results with different initial values of parameters. In the second set of experiments, we examined the relationship between model types and their parsing performances. In the third set of experiments, we compared the three parsing methods described in the previous section. Finally, we show the result of a parsing experiment using the standard test set.

We used sections 2 through 20 of the Penn WSJ corpus as training data and section 21 as heldout data. The heldout data was used for early stopping; i.e., the estimation was stopped when the rate

- If $i_1 \in I$ is not a pre-terminal node, for each $\eta = i_2 i_3 \in \delta(i_1)$, let $A, B,$ and C be non-terminal symbols of $i_1, i_2,$ and i_3. Then,
$$q(i_1 \to \eta) = \frac{\sum_{x \in H} \sum_{y \in H} \sum_{z \in H} P_{\text{out}}(i_1[x]) \beta(A[x] \to B[y]C[z]) P_{\text{in}}(i_2[y]) P_{\text{in}}(i_3[z])}{\sum_{x \in H} P_{\text{out}}(i_1[x]) P_{\text{in}}(i_1[x])}.$$
- If $i \in I$ is a pre-terminal node above word w_k, then $q(i \to w_k) = 1$.
- If $i \in I$ is a root node, let A be the non-terminal symbol of i. Then $q_r(i) = \frac{1}{P(w)} \sum_{x \in H} \pi(A[x]) P_{\text{in}}(i[x])$.

Figure 4: Optimal parameters of approximate distribution Q.

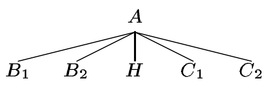


Figure 5: Original subtree.

	1	2	3	4	average $\pm\ \sigma$
training LL	-115	-114	-115	-114	-114 ± 0.41
heldout LL	-114	-115	-115	-114	-114 ± 0.29
LR	86.7	86.3	86.3	87.0	86.6 ± 0.27
LP	86.2	85.6	85.5	86.6	86.0 ± 0.48

Table 1: Dependency on initial values.

of increase in the likelihood of the heldout data became lower than a certain threshold. Section 22 was used as test data in all parsing experiments except in the final one, in which section 23 was used. We stripped off all function tags and eliminated empty nodes in the training and heldout data, but any other pre-processing, such as comma raising or base-NP marking (Collins, 1999), was not done except for binarizations.

4.1 Dependency on initial values

To see the degree of dependency of trained models on initializations, four instances of the same model were trained with different initial values of parameters.[3] The model used in this experiment was created by CENTER-PARENT binarization and $|H|$ was set to 16. Table 1 lists training/heldout data log-likelihood per sentence (LL) for the four instances and their parsing performances on the test set (section 22). The parsing performances were obtained using the approximate distribution method in Section 3.2. Different initial values were shown to affect the results of training to some extent (Table 1).

[3]The initial value for an annotated rule probability, $\beta(A[x] \to B[y]C[z])$, was created by randomly multiplying the maximum likelihood estimation of the corresponding PCFG rule probability, $P(A \to BC)$, as follows:

$$\beta(A[x] \to B[y]C[z]) = Z_A^{-1} e^{\gamma} P(A \to BC),$$

where γ is a random number that is uniformly distributed in $[-\log 3, \log 3]$ and Z_A is a normalization constant.

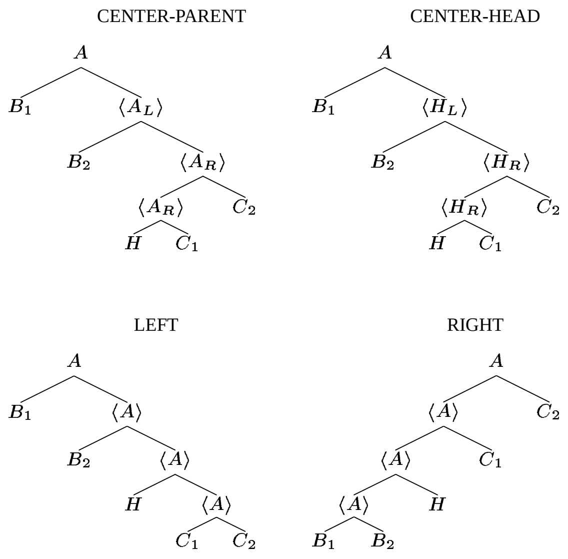


Figure 6: Four types of binarization (H: head daughter).

4.2 Model types and parsing performance

We compared four types of binarization. The original form is depicted in Figure 5 and the results are shown in Figure 6. In the first two methods, called CENTER-PARENT and CENTER-HEAD, the head-finding rules of Collins (1999) were used. We obtained an observable grammar R for each model by reading off grammar rules from the binarized training trees. For each binarization method, PCFG-LA models with different numbers of latent annotation symbols, $|H| = 1, 2, 4, 8,$ and 16, were trained.

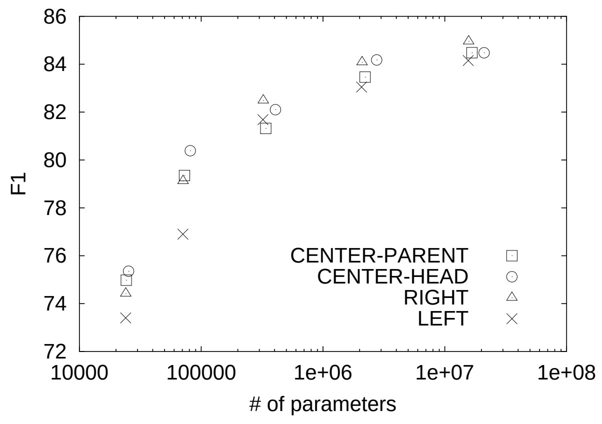


Figure 7: Model size vs. parsing performance.

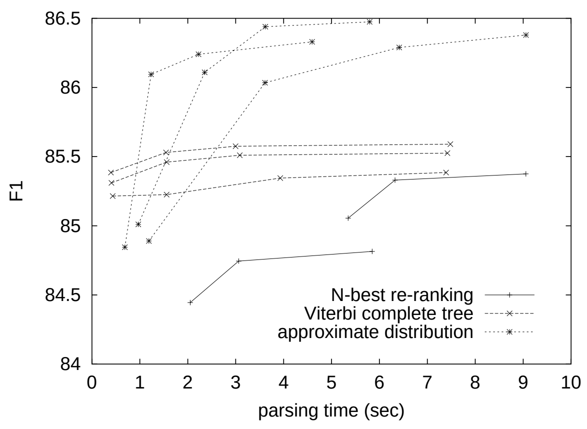


Figure 8: Comparison of parsing methods.

The relationships between the number of parameters in the models and their parsing performances are shown in Figure 7. Note that models created using different binarization methods have different numbers of parameters for the same $|H|$. The parsing performances were measured using F_1 scores of the parse trees that were obtained by re-ranking of 1000-best parses by a PCFG.

We can see that the parsing performance gets better as the model size increases. We can also see that models of roughly the same size yield similar performances regardless of the binarization scheme used for them, except the models created using `LEFT` binarization with small numbers of parameters ($|H| =$ 1 and 2). Taking into account the dependency on initial values at the level shown in the previous experiment, we cannot say that any single model is superior to the other models when the sizes of the models are large enough.

The results shown in Figure 7 suggest that we could further improve parsing performance by increasing the model size. However, both the memory size and the training time are more than linear in $|H|$, and the training time for the largest ($|H| = 16$) models was about 15 hours for the models created using `CENTER-PARENT`, `CENTER-HEAD`, and `LEFT` and about 20 hours for the model created using `RIGHT`. To deal with larger (e.g., $|H|$ = 32 or 64) models, we therefore need to use a model search that reduces the number of parameters while maintaining the model's performance, and an approximation during training to reduce the training time.

4.3 Comparison of parsing methods

The relationships between the average parse time and parsing performance using the three parsing methods described in Section 3 are shown in Figure 8. A model created using `CENTER-PARENT` with $|H| = 16$ was used throughout this experiment.

The data points were made by varying configurable parameters of each method, which control the number of candidate parses. To create the candidate parses, we first parsed input sentences using a PCFG[4], using beam thresholding with beam width α. The data points on a line in the figure were created by varying α with other parameters fixed. The first method re-ranked the N-best parses enumerated from the chart after the PCFG parsing. The two lines for the first method in the figure correspond to N = 100 and N = 300. In the second and the third methods, we removed all the dominance relations among chart items that did not contribute to any parses whose PCFG-scores were higher than $\gamma P_{\max}$, where $P_{\max}$ is the PCFG-score of the best parse in the chart. The parses remaining in the chart were the candidate parses for the second and the third methods. The different lines for the second and the third methods correspond to different values of γ.

The third method outperforms the other two methods unless the parse time is very limited (i.e., < 1

[4]The PCFG used in creating the candidate parses is roughly the same as the one that Klein and Manning (2003) call a 'markovised PCFG with vertical order = 2 and horizontal order = 1' and was extracted from Section 02-20. The PCFG itself gave a performance of 79.6/78.5 LP/LR on the development set. This PCFG was also used in the experiment in Section 4.4.

$\leq$ 40 words	LR	LP	CB	0 CB
This paper	86.7	86.6	1.19	61.1
Klein and Manning (2003)	85.7	86.9	1.10	60.3
Collins (1999)	88.5	88.7	0.92	66.7
Charniak (1999)	90.1	90.1	0.74	70.1
$\leq$ 100 words	LR	LP	CB	0 CB
This paper	86.0	86.1	1.39	58.3
Klein and Manning (2003)	85.1	86.3	1.31	57.2
Collins (1999)	88.1	88.3	1.06	64.0
Charniak (1999)	89.6	89.5	0.88	67.6

Table 2: Comparison with other parsers.

sec is required), as shown in the figure. The superiority of the third method over the first method seems to stem from the difference in the number of candidate parses from which the outputs are selected.[5] The superiority of the third method over the second method is a natural consequence of the consistent use of $P(T)$ both in the estimation (as the objective function) and in the parsing (as the score of a parse).

4.4 Comparison with related work

Parsing performance on section 23 of the WSJ corpus using a PCFG-LA model is shown in Table 2. We used the instance of the four compared in the second experiment that gave the best results on the development set. Several previously reported results on the same test set are also listed in Table 2.

Our result is lower than the state-of-the-art lexicalized PCFG parsers (Collins, 1999; Charniak, 1999), but comparable to the unlexicalized PCFG parser of Klein and Manning (2003). Klein and Manning's PCFG is annotated by many linguistically motivated features that they found using extensive manual feature selection. In contrast, our method induces all parameters automatically, except that manually written head-rules are used in binarization. Thus, our method can extract a considerable amount of hidden regularity from parsed corpora. However, our result is worse than the lexicalized parsers despite the fact that our model has access to words in the sentences. It suggests that certain types of information used in those lexicalized parsers are hard to be learned by our approach.

[5] Actually, the number of parses contained in the packed forest is more than 1 million for over half of the test sentences when $\alpha = 10^{-4}$ and $\gamma = 10^{-3}$, while the number of parses for which the first method can compute the exact probability in a comparable time (around 4 sec) is only about 300.

References

Eugene Charniak. 1999. A maximum-entropy-inspired parser. Technical Report CS-99-12.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proc. COLING*, pages 183–189.

Stephen Clark and James R. Curran. 2004. Parsing the wsj using ccg and log-linear models. In *Proc. ACL*, pages 104–111.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Brendan J. Frey, Relu Patrascu, Tommi Jaakkola, and Jodi Moran. 2000. Sequentially fitting "inclusive" trees for inference in noisy-OR networks. In *Proc. NIPS*, pages 493–499.

Joshua Goodman. 1996a. Efficient algorithms for parsing the DOP model. In *Proc. EMNLP*, pages 143–152.

Joshua Goodman. 1996b. Parsing algorithms and metric. In *Proc. ACL*, pages 177–183.

Joshua Goodman. 1997. Probabilistic feature grammars. In *Proc. IWPT*.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proc. HLT-NAACL*, pages 103–110.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. ACL*, pages 423–430.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. ACL*, pages 128–135.

Libin Shen. 2004. Nondeterministic LTAG derivation tree extraction. In *Proc. TAG+7*, pages 199–203.

Khalil Simaán. 2002. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151.

Khalil Simaán. 2003. On maximizing metrics for syntactic disambiguation. In *Proc. IWPT*.

Takehito Utsuro, Syuuji Kodama, and Yuji Matsumoto. 1996. Generalization/specialization of context free grammars based-on entropy of non-terminals. In *Proc. JSAI (in Japanese)*, pages 327–330.

Probabilistic disambiguation models for wide-coverage HPSG parsing

Yusuke Miyao
Department of Computer Science
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
yusuke@is.s.u-tokyo.ac.jp

Jun'ichi Tsujii
Department of Computer Science
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
CREST, JST
tsujii@is.s.u-tokyo.ac.jp

Abstract

This paper reports the development of log-linear models for the disambiguation in wide-coverage HPSG parsing. The estimation of log-linear models requires high computational cost, especially with wide-coverage grammars. Using techniques to reduce the estimation cost, we trained the models using 20 sections of Penn Treebank. A series of experiments empirically evaluated the estimation techniques, and also examined the performance of the disambiguation models on the parsing of real-world sentences.

1 Introduction

Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) has been studied extensively from both linguistic and computational points of view. However, despite research on HPSG processing efficiency (Oepen et al., 2002a), the application of HPSG parsing is still limited to specific domains and short sentences (Oepen et al., 2002b; Toutanova and Manning, 2002). Scaling up HPSG parsing to assess real-world texts is an emerging research field with both theoretical and practical applications.

Recently, a wide-coverage grammar and a large treebank have become available for English HPSG (Miyao et al., 2004). A large treebank can be used as training and test data for statistical models. Therefore, we now have the basis for the development and the evaluation of statistical disambiguation models for wide-coverage HPSG parsing.

The aim of this paper is to report the development of log-linear models for the disambiguation in wide-coverage HPSG parsing, and their empirical evaluation through the parsing of the Wall Street Journal of Penn Treebank II (Marcus et al., 1994). This is challenging because the estimation of log-linear models is computationally expensive, and we require solutions to make the model estimation tractable. We apply two techniques for reducing the training cost. One is the estimation on a packed representation of HPSG parse trees (Section 3). The other is the filtering of parse candidates according to a preliminary probability distribution (Section 4).

To our knowledge, this work provides the first results of extensive experiments of parsing Penn Treebank with a probabilistic HPSG. The results from the Wall Street Journal are significant because the complexity of the sentences is different from that of short sentences. Experiments of the parsing of real-world sentences can properly evaluate the effectiveness and possibility of parsing models for HPSG.

2 Disambiguation models for HPSG

Discriminative log-linear models are now becoming a de facto standard for probabilistic disambiguation models for deep parsing (Johnson et al., 1999; Riezler et al., 2002; Geman and Johnson, 2002; Miyao and Tsujii, 2002; Clark and Curran, 2004b; Kaplan et al., 2004). Previous studies on probabilistic models for HPSG (Toutanova and Manning, 2002; Baldridge and Osborne, 2003; Malouf and van Noord, 2004) also adopted log-linear models. HPSG exploits feature structures to represent linguistic constraints. Such constraints are known to introduce inconsistencies in probabilistic models estimated using simple relative frequency (Abney, 1997). Log-linear models are required for credible probabilistic models and are also beneficial for incorporating various overlapping features.

This study follows previous studies on the probabilistic models for HPSG. The probability, $p(t|s)$, of producing the parse result t from a given sentence s is defined as

$$p(t|s) = \frac{1}{Z_s} p_0(t|s) \exp(\sum_i f_i(t,s)\lambda_i(t,s))$$

$$Z_s = \sum_{t' \in T(s)} p_0(t'|s) \exp(\sum_i f_i(t',s)\lambda_i(t',s)),$$

where $p_0(t|s)$ is a reference distribution (usually assumed to be a uniform distribution), and $T(s)$ is a set of parse candidates assigned to s. The feature function $f_i(t,s)$ represents the characteristics of t and s, while the corresponding model parameter $\lambda_i(t,s)$ is its weight. Model parameters that maximize the log-likelihood of the training data are computed using a numerical optimization method (Malouf, 2002).

Estimation of the above model requires a set of pairs $\langle t_s, T(s)\rangle$, where t_s is the correct parse for sentence s. While t_s is provided by a treebank, $T(s)$ is computed by parsing each s in the treebank. Previous studies assumed $T(s)$ could be enumerated; however, the assumption is impractical because the size of $T(s)$ is exponentially related to the length of s. The problem of exponential explosion is inevitable in the wide-coverage parsing of real-world texts because many parse candidates are produced to support various constructions in long sentences.

3 Packed representation of HPSG parse trees

To avoid exponential explosion, we represent $T(s)$ in a packed form of HPSG parse trees. A parse tree of HPSG is represented as a set of tuples $\langle m, l, r\rangle$, where m, l, and r are the signs of mother, left daughter, and right daughter, respectively[1]. In chart parsing, partial parse candidates are stored in a *chart*, in which phrasal signs are identified and packed into an equivalence class if they are determined to be equivalent and dominate the same word sequence. A set

[1]For simplicity, only binary trees are considered. Extension to unary and n-ary ($n > 2$) trees is trivial.

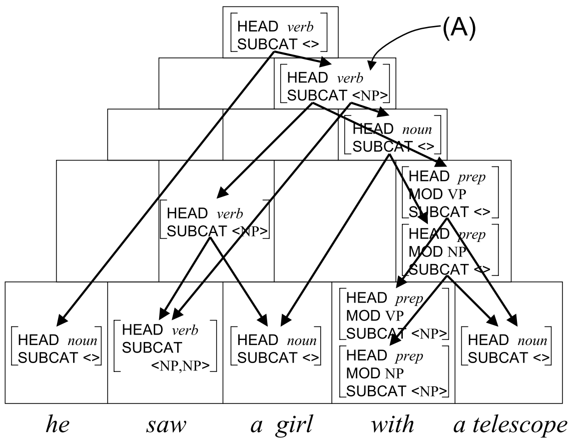


Figure 1: Chart for parsing *"he saw a girl with a telescope"*

of parse trees is then represented as a set of relations among equivalence classes.

Figure 1 shows a chart for parsing *"he saw a girl with a telescope"*, where the modifiee (*"saw"* or *"girl"*) of *"with"* is ambiguous. Each feature structure expresses an equivalence class, and the arrows represent immediate-dominance relations. The phrase, *"saw a girl with a telescope"*, has two trees (A in the figure). Since the signs of the top-most nodes are equivalent, they are packed into an equivalence class. The ambiguity is represented as two pairs of arrows that come out of the node.

Formally, a set of HPSG parse trees is represented in a chart as a tuple $\langle E, E_r, \alpha\rangle$, where E is a set of equivalence classes, $E_r \subseteq E$ is a set of root nodes, and $\alpha : E \to 2^{E\times E}$ is a function to represent immediate-dominance relations.

Our representation of the chart can be interpreted as an instance of *a feature forest* (Miyao and Tsujii, 2002; Geman and Johnson, 2002). A feature forest is an "and/or" graph to represent exponentially-many tree structures in a packed form. If $T(s)$ is represented in a feature forest, $p(t|T(s))$ can be estimated using dynamic programming without unpacking the chart. A feature forest is formally defined as a tuple, $\langle C, D, R, \gamma, \delta\rangle$, where C is a set of conjunctive nodes, D is a set of disjunctive nodes, $R \subseteq C$ is a set of root nodes[2], $\gamma : D \to 2^C$ is a conjunctive daughter function, and $\delta : C \to 2^D$ is a disjunctive

[2]For the ease of explanation, the definition of root node is slightly different from the original.

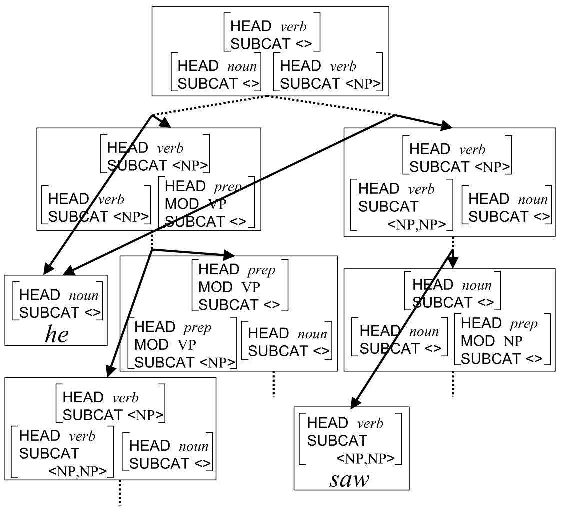


Figure 2: Packed representation of HPSG parse trees in Figure 1

daughter function. The feature functions $f_i(t, s)$ are assigned to conjunctive nodes.

The simplest way to map a chart of HPSG parse trees into a feature forest is to map each equivalence class $e \in E$ to a conjunctive node $c \in C$. However, in HPSG parsing, important features for disambiguation are combinations of a mother and its daughters, i.e., $\langle m, l, r \rangle$. Hence, we map the tuple $\langle e_m, e_l, e_r \rangle$, which corresponds to $\langle m, l, r \rangle$, into a conjunctive node.

Figure 2 shows (a part of) the HPSG parse trees in Figure 1 represented as a feature forest. Square boxes are conjunctive nodes, dotted lines express a disjunctive daughter function, and solid arrows represent a conjunctive daughter function.

The mapping is formally defined as follows.

- $C = \{\langle e_m, e_l, e_r \rangle | e_m \in E \wedge e_l, e_r \in E. \langle e_l, e_r \rangle \in \alpha(e_m)\}$,
- $D = E$,
- $R = \{\langle e_m, e_l, e_r \rangle | e_m \in E_r \wedge \langle e_m, e_l, e_r \rangle \in C\}$,
- $\gamma = \{\langle e_m, C(e_m) \rangle | e_m \in E \wedge C(e_m) = \{\langle e_m, e_l, e_r \rangle | e_l, e_r \in E. \langle e_l, e_r \rangle \in \alpha(e_m)\}\}$, and
- $\delta = \{\langle \langle e_m, e_l, e_r \rangle, \{e_l, e_r\} \rangle | \langle e_m, e_l, e_r \rangle \in C\}$.

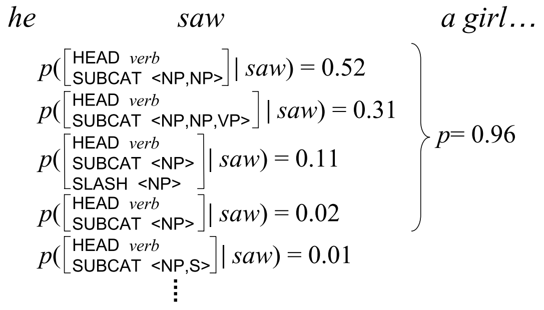

he
saw
a girl…
p(HEAD verb SUBCAT <NP,NP> | saw) = 0.52
p(HEAD verb SUBCAT <NP,NP,VP> | saw) = 0.31
p(HEAD verb SUBCAT <NP> SLASH <NP> | saw) = 0.11
p(HEAD verb SUBCAT <NP> | saw) = 0.02
p(HEAD verb SUBCAT <NP,S> | saw) = 0.01
p= 0.96

Figure 3: Filtering of lexical entries for *"saw"*

4 Filtering by preliminary distribution

The above method allows for the tractable estimation of log-linear models on exponentially-many HPSG parse trees. However, despite the development of methods to improve HPSG parsing efficiency (Oepen et al., 2002a), the exhaustive parsing of all sentences in a treebank is still expensive.

Our idea is that we can omit the computation of parse trees with low probabilities in the estimation stage because $T(s)$ can be approximated with parse trees with high probabilities. To achieve this, we first prepared *a preliminary probabilistic model* whose estimation did not require the parsing of a treebank. The preliminary model was used to reduce the search space for parsing a training treebank.

The preliminary model in this study is a unigram model, $\bar{p}(t|s) = \prod_{w \in s} p(l|w)$, where $w \in s$ is a word in the sentence s, and l is a lexical entry assigned to w. This model can be estimated without parsing a treebank.

Given this model, we restrict the number of lexical entries used to parse a treebank. With a threshold n for the number of lexical entries and a threshold ϵ for the probability, lexical entries are assigned to a word in descending order of probability, until the number of assigned entries exceeds n, or the accumulated probability exceeds ϵ. If the lexical entry necessary to produce the correct parse is not assigned, it is additionally assigned to the word.

Figure 3 shows an example of filtering lexical entries assigned to *"saw"*. With $\epsilon = 0.95$, four lexical entries are assigned. Although the lexicon includes other lexical entries, such as a verbal entry taking a sentential complement ($p = 0.01$ in the figure), they are filtered out. This method reduces the time for

RULE	the name of the applied schema
DIST	the distance between the head words of the daughters
COMMA	whether a comma exists between daughters and/or inside of daughter phrases
SPAN	the number of words dominated by the phrase
SYM	the symbol of the phrasal category (e.g. NP, VP)
WORD	the surface form of the head word
POS	the part-of-speech of the head word
LE	the lexical entry assigned to the head word

Table 1: Templates of atomic features

parsing a treebank, while this approximation causes bias in the training data and results in lower accuracy. The trade-off between the parsing cost and the accuracy will be examined experimentally.

We have several ways to integrate $\bar{p}$ with the estimated model $p(t|T(s))$. In the experiments, we will empirically compare the following methods in terms of accuracy and estimation time.

Filtering only The unigram probability $\bar{p}$ is used only for filtering.

Product The probability is defined as the product of $\bar{p}$ and the estimated model p.

Reference distribution $\bar{p}$ is used as a reference distribution of p.

Feature function $\log \bar{p}$ is used as a feature function of p. This method was shown to be a generalization of the reference distribution method (Johnson and Riezler, 2000).

5 Features

Feature functions in the log-linear models are designed to capture the characteristics of $\langle e_m, e_l, e_r \rangle$. In this paper, we investigate combinations of the atomic features listed in Table 1. The following combinations are used for representing the characteristics of the binary/unary schema applications.

$$f_{\text{binary}} = \left\langle \begin{array}{l} \text{RULE,DIST,COMMA,} \\ \text{SPAN}_l, \text{SYM}_l, \text{WORD}_l, \text{POS}_l, \text{LE}_l, \\ \text{SPAN}_r, \text{SYM}_r, \text{WORD}_r, \text{POS}_r, \text{LE}_r \end{array} \right\rangle$$

$$f_{\text{unary}} = \langle \text{RULE,SYM,WORD,POS,LE} \rangle$$

In addition, the following is for expressing the condition of the root node of the parse tree.

$$f_{\text{root}} = \langle \text{SYM,WORD,POS,LE} \rangle$$

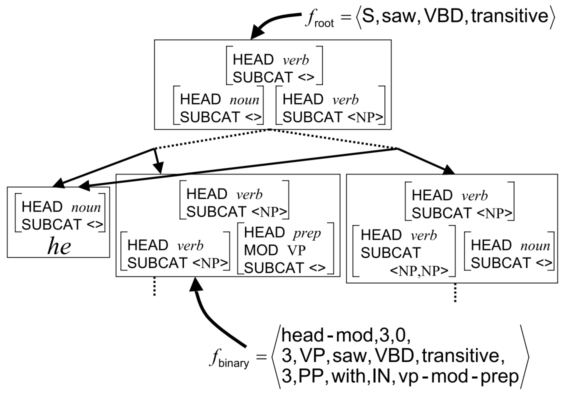

$f_{\text{root}} = \langle$S, saw, VBD, transitive$\rangle$
HEAD verb
SUBCAT <>
HEAD noun
SUBCAT <>
HEAD verb
SUBCAT <NP>
he
HEAD prep
MOD VP
SUBCAT <>
SUBCAT <NP,NP>
$f_{\text{binary}} = \langle$head-mod,3,0, 3,VP,saw,VBD,transitive, 3,PP,with,IN,vp-mod-prep$\rangle$

Figure 4: Example features

Figure 4 shows examples: f_{root} is for the root node, in which the phrase symbol is S and the surface form, part-of-speech, and lexical entry of the lexical head are *"saw"*, VBD, and a transitive verb, respectively. f_{binary} is for the binary rule application to *"saw a girl"* and *"with a telescope"*, in which the applied schema is the Head-Modifier Schema, the left daughter is VP headed by *"saw"*, and the right daughter is PP headed by *"with"*, whose part-of-speech is IN and the lexical entry is a VP-modifying preposition.

In an actual implementation, some of the atomic features are abstracted (i.e., ignored) for smoothing. Table 2 shows a full set of templates of combined features used in the experiments. Each row represents a template of a feature function. A check means the atomic feature is incorporated while a hyphen means the feature is ignored.

Restricting the domain of feature functions to $\langle e_m, e_l, e_r \rangle$ seems to limit the flexibility of feature design. Although it is true to some extent, this does not necessarily mean the impossibility of incorporating features on nonlocal dependencies into the model. This is because a feature forest model does not assume probabilistic independence of conjunctive nodes. This means that we can unpack a part of the forest without changing the model. Actually, in our previous study (Miyao et al., 2003), we successfully developed a probabilistic model including features on nonlocal predicate-argument dependencies. However, since we could not observe significant improvements by incorporating nonlocal features, this paper investigates only the features described above.

RULE	DIST	COMMA	SPAN	SYM	WORD	POS	LE
√	√	√	–	–	√	√	√
√	√	√	–	–	√	√	–
√	√	√	–	–	√	–	√
√	√	√	–	√	√	–	–
√	–	√	√	–	√	√	√
√	–	√	√	–	√	√	–
√	–	√	√	–	√	–	√
√	–	√	√	√	√	–	–
√	√	√	–	–	–	√	√
√	√	√	–	–	–	√	–
√	√	√	–	–	–	–	√
√	√	√	–	√	–	–	–
√	–	√	√	–	–	√	√
√	–	√	√	–	–	√	–
√	–	√	√	–	–	–	√
√	–	√	√	√	–	–	–

RULE	SYM	WORD	POS	LE
√	–	√	√	√
√	–	√	√	–
√	–	√	–	√
√	√	√	–	–
√	–	–	√	√
√	–	–	√	–
√	–	–	–	√
√	√	–	–	–

SYM	WORD	POS	LE
–	√	√	√
–	√	√	–
–	√	–	√
√	√	–	–
–	–	√	√
–	–	√	–
–	–	–	√
√	–	–	–

Table 2: Feature templates for binary schema (left), unary schema (center), and root condition (right)

	Avg. length	LP	LR	UP	UR	F-score
Section 22 (< 40 words)	20.69	87.18	86.23	90.67	89.68	86.70
Section 22 (< 100 words)	22.43	86.99	84.32	90.45	87.67	85.63
Section 23 (< 40 words)	20.52	87.12	85.45	90.65	88.91	86.27
Section 23 (< 100 words)	22.23	86.81	84.64	90.29	88.03	85.71

Table 3: Accuracy for development/test sets

6 Experiments

We used an HPSG grammar derived from Penn Treebank (Marcus et al., 1994) Section 02-21 (39,832 sentences) by our method of grammar development (Miyao et al., 2004). The training data was the HPSG treebank derived from the same portion of the Penn Treebank[3]. For the training, we eliminated sentences with no less than 40 words and for which the parser could not produce the correct parse. The resulting training set consisted of 33,574 sentences. The treebanks derived from Sections 22 and 23 were used as the development (1,644 sentences) and final test sets (2,299 sentences). We measured the accuracy of predicate-argument dependencies output by the parser. A dependency is defined as a tuple $\langle \pi, w_h, a, w_a \rangle$, where π is the predicate type (e.g., adjective, intransitive verb), w_h is the head word of the predicate, a is the argument label (MODARG, ARG1, ..., ARG4), and w_a is the head word of the argument. Labeled precision/recall (LP/LR) is the ratio of tuples correctly identified by the parser, while unlabeled precision/recall (UP/UR) is the ratio of w_h and w_a correctly identified regardless of π and a. The F-score is the harmonic mean of LP and LR. The accuracy was measured by parsing test sentences with part-of-speech tags provided by the treebank. The Gaussian prior was used for smoothing (Chen and Rosenfeld, 1999), and its hyper-parameter was tuned for each model to maximize the F-score for the development set. The optimization algorithm was the limited-memory BFGS method (Nocedal and Wright, 1999). All the following experiments were conducted on AMD Opteron servers with a 2.0-GHz CPU and 12-GB memory.

[3]The programs to make the grammar and the treebank from Penn Treebank are available at `http://www-tsujii.is.s.u-tokyo.ac.jp/enju/`.

Table 3 shows the accuracy for the development/test sets. Features occurring more than twice were included in the model (598,326 features). Filtering was done by the reference distribution method with $n = 10$ and $\epsilon = 0.95$. The unigram model for filtering was a log-linear model with two feature templates, $\langle$WORD, POS, LE$\rangle$ and $\langle$POS, LE$\rangle$ (24,847 features). Our results cannot be strictly compared with other grammar formalisms because each formalism represents predicate-argument dependencies differently; for reference, our results are competitive with the corresponding measures reported for Combinatory Categorial Grammar (CCG) (LP/LR = 86.6/86.3) (Clark and Curran, 2004b). Different from the results of CCG and PCFG (Collins, 1999; Charniak, 2000), the recall was clearly lower than precision. This results from the HPSG grammar having stricter feature constraints and the parser not being able to produce parse results for around one percent of the sentences. To improve recall, we need techniques of robust processing with HPSG.

	LP	LR	Estimation time (sec.)
Filtering only	34.90	23.34	702
Product	86.71	85.55	1,758
Reference dist.	87.12	85.45	655
Feature function	84.89	83.06	1,203

Table 4: Estimation method vs. accuracy and estimation time

n, ϵ	F-score	Estimation time (sec.)	Parsing time (sec.)	Memory usage (MB)
5, 0.80	84.31	161	7,827	2,377
5, 0.90	84.69	207	9,412	2,992
5, 0.95	84.70	240	12,027	3,648
5, 0.98	84.81	340	15,168	4,590
10, 0.80	84.79	164	8,858	2,658
10, 0.90	85.77	298	13,996	4,062
10, 0.95	86.27	654	25,308	6,324
10, 0.98	86.56	1,778	55,691	11,700
15, 0.80	84.68	180	9,337	2,676
15, 0.90	85.85	308	14,915	4,220
15, 0.95	86.68	854	32,757	7,766

Table 5: Filtering threshold vs. accuracy and estimation time

Table 4 compares the estimation methods introduced in Section 4. In all of the following experiments, we show the accuracy for the test set ($<$ 40 words) only. Table 4 revealed that our simple method of filtering caused a fatal bias in training data when a preliminary distribution was used only for filtering. However, the model combined with a preliminary model achieved sufficient accuracy. The reference distribution method achieved higher accuracy and lower cost. The feature function method achieved lower accuracy in our experiments. A possible reason is that a hyper-parameter of the prior was set to the same value for all the features including the feature of the preliminary distribution.

Table 5 shows the results of changing the filtering threshold. We can determine the correlation between the estimation/parsing cost and accuracy. In our experiment, $n \geq 10$ and $\epsilon \geq 0.90$ seem necessary to preserve the F-score over 85.0.

Figure 5 shows the accuracy for each sentence length. It is apparent from this figure that the accuracy was significantly higher for shorter sentences ($<$ 10 words). This implies that experiments with only short sentences overestimate the performance of parsers. Sentences with at least 10 words are necessary to properly evaluate the performance of parsing real-world texts.

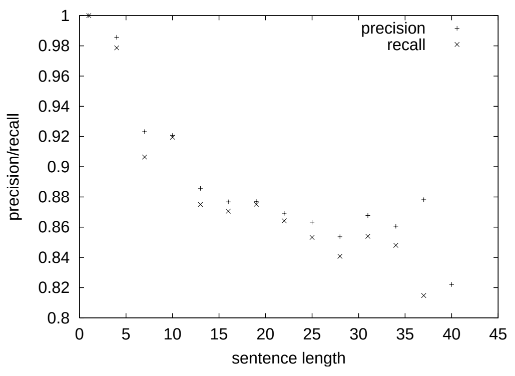


Figure 5: Sentence length vs. accuracy

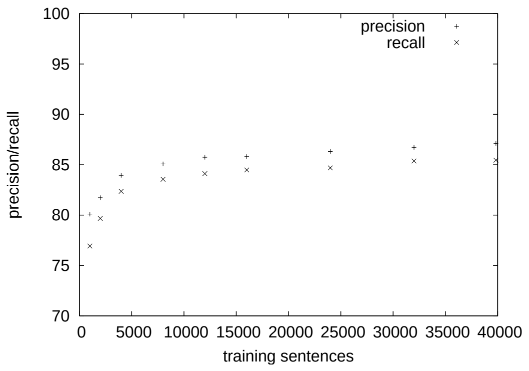


Figure 6: Corpus size vs. accuracy

Figure 6 shows the learning curve. A feature set was fixed, while the parameter of the prior was optimized for each model. High accuracy was attained even with small data, and the accuracy seemed to be saturated. This indicates that we cannot further improve the accuracy simply by increasing training data. The exploration of new types of features is necessary for higher accuracy.

Table 6 shows the accuracy with difference feature sets. The accuracy was measured by removing some of the atomic features from the final model. The last row denotes the accuracy attained by the preliminary model. The numbers in bold type represent that the difference from the final model was significant according to stratified shuffling tests (Cohen, 1995) with p-value < 0.05. The results indicate that DIST, COMMA, SPAN, WORD, and POS features contributed to the final accuracy, although the dif-

Features	LP	LR	# features
All	87.12	85.45	623,173
−RULE	86.98	85.37	620,511
−DIST	**86.74**	**85.09**	603,748
−COMMA	**86.55**	**84.77**	608,117
−SPAN	**86.53**	**84.98**	583,638
−SYM	86.90	85.47	614,975
−WORD	**86.67**	**84.98**	116,044
−POS	**86.36**	**84.71**	430,876
−LE	87.03	85.37	412,290
−DIST,SPAN	**85.54**	**84.02**	294,971
−DIST,SPAN, COMMA	**83.94**	**82.44**	286,489
−RULE,DIST, SPAN,COMMA	**83.61**	**81.98**	283,897
−WORD,LE	**86.48**	**84.91**	50,258
−WORD,POS	**85.56**	**83.94**	64,915
−WORD,POS,LE	**84.89**	**83.43**	33,740
−SYM,WORD, POS,LE	**82.81**	**81.48**	26,761
None	**78.22**	**76.46**	24,847

Table 6: Accuracy with different feature sets

Error cause	# of errors
Argument/modifier distinction	58
temporal noun	21
to-infinitive	15
others	22
Attachment	53
prepositional phrase	18
to-infinitive	10
relative clause	8
others	17
Lexical ambiguity	42
participle/adjective	15
preposition/modifier	14
others	13
Comma	19
Coordination	14
Noun phrase identification	13
Zero-pronoun resolution	9
Others	17

Table 7: Error analysis

ferences were slight. In contrast, RULE, SYM, and LE features did not affect the accuracy. However, if each of them was removed together with another feature, the accuracy decreased drastically. This implies that such features had overlapping information.

Table 7 shows the manual classification of the causes of errors in 100 sentences randomly chosen from the development set. In our evaluation, one error source may cause multiple errors of dependencies. For example, if a wrong lexical entry was assigned to a verb, all the argument dependencies of the verb are counted as errors. The numbers in the table include such double-counting. Major causes were classified into three types: argument/modifier distinction, attachment ambiguity, and lexical ambiguity. While attachment/lexical ambiguities are well-known causes, the other is peculiar to deep parsing. Most of the errors cannot be resolved by features we investigated in this study, and the design of other features is crucial for further improvements.

7 Discussion and related work

Experiments on deep parsing of Penn Treebank have been reported for Combinatory Categorial Grammar (CCG) (Clark and Curran, 2004b) and Lexical Functional Grammar (LFG) (Kaplan et al., 2004). They developed log-linear models on a packed representation of parse forests, which is similar to our representation. Although HPSG exploits further complicated feature constraints and requires high computational cost, our work has proved that log-linear models can be applied to HPSG parsing and attain accurate and wide-coverage parsing.

Clark and Curran (2004a) described a method of reducing the cost of parsing a training treebank in the context of CCG parsing. They first assigned to each word a small number of *supertags*, which correspond to lexical entries in our case, and parsed *supertagged sentences*. Since they did not mention the probabilities of supertags, their method corresponds to our "filtering only" method. However, they also applied the same supertagger in a parsing stage, and this seemed to be crucial for high accuracy. This means that they estimated the probability of producing a parse tree from a supertagged sentence.

Another approach to estimating log-linear models for HPSG is to extract a small *informative sample* from the original set $T(s)$ (Osborne, 2000). Malouf and van Noord (2004) successfully applied this method to German HPSG. The problem with this method was in the approximation of exponentially many parse trees by a polynomial-size sample. However, their method has the advantage that any features on a parse tree can be incorporated into the model. The trade-off between approximation and locality of features is an outstanding problem.

Other discriminative classifiers were applied to the disambiguation in HPSG parsing (Baldridge and Osborne, 2003; Toutanova et al., 2004). The problem of exponential explosion is also inevitable for their methods. An approach similar to ours may be applied to them, following the study on the learning of a discriminative classifier for a packed representation (Taskar et al., 2004).

As discussed in Section 6, exploration of other features is indispensable to further improvements. A possible direction is to encode larger contexts of parse trees, which were shown to improve the accuracy (Toutanova and Manning, 2002; Toutanova et al., 2004). Future work includes the investigation of such features, as well as the abstraction of lexical dependencies like semantic classes.

References

S. P. Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4).

J. Baldridge and M. Osborne. 2003. Active learning for HPSG parse selection. In *CoNLL-03*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. NAACL-2000*, pages 132–139.

S. Chen and R. Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University.

S. Clark and J. R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proc. COLING-04*.

S. Clark and J. R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proc. 42th ACL*.

P. R. Cohen. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Univ. of Pennsylvania.

S. Geman and M. Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proc. 40th ACL*.

M. Johnson and S. Riezler. 2000. Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proc. 1st NAACL*.

M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proc. ACL'99*, pages 535–541.

R. M. Kaplan, S. Riezler, T. H. King, J. T. Maxwell III, and A. Vasserman. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proc. HLT/NAACL'04*.

R. Malouf and G. van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *Proc. IJCNLP-04 Workshop "Beyond Shallow Analyses"*.

R. Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. CoNLL-2002*.

M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

Y. Miyao and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proc. HLT 2002*.

Y. Miyao, T. Ninomiya, and J. Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proc. RANLP 2003*, pages 285–291.

Y. Miyao, T. Ninomiya, and J. Tsujii. 2004. Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proc. IJCNLP-04*.

J. Nocedal and S. J. Wright. 1999. *Numerical Optimization*. Springer.

S. Oepen, D. Flickinger, J. Tsujii, and H. Uszkoreit, editors. 2002a. *Collaborative Language Engineering: A Case Study in Efficient Grammar-Based Processing*. CSLI Publications.

S. Oepen, K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants. 2002b. The LinGO, Redwoods treebank. motivation and preliminary applications. In *Proc. COLING 2002*.

M. Osborne. 2000. Estimation of stochastic attribute-value grammar using an informative sample. In *Proc. COLING 2000*.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

S. Riezler, T. H. King, R. M. Kaplan, R. Crouch, J. T. Maxwell III, and M. Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proc. 40th ACL*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *EMNLP 2004*.

K. Toutanova and C. D. Manning. 2002. Feature selection for a rich HPSG grammar using decision trees. In *Proc. CoNLL-2002*.

K. Toutanova, P. Markova, and C. Manning. 2004. The leaf projection path view of parse trees: Exploring string kernels for HPSG parse selection. In *EMNLP 2004*.

Online Large-Margin Training of Dependency Parsers

Ryan McDonald **Koby Crammer** **Fernando Pereira**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA
{ryantm,crammer,pereira}@cis.upenn.edu

Abstract

We present an effective training algorithm for linearly-scored dependency parsers that implements online large-margin multi-class training (Crammer and Singer, 2003; Crammer et al., 2003) on top of efficient parsing techniques for dependency trees (Eisner, 1996). The trained parsers achieve a competitive dependency accuracy for both English and Czech with no language specific enhancements.

1 Introduction

Research on training parsers from annotated data has for the most part focused on models and training algorithms for phrase structure parsing. The best phrase-structure parsing models represent generatively the joint probability $P(\boldsymbol{x}, \boldsymbol{y})$ of sentence $\boldsymbol{x}$ having the structure $\boldsymbol{y}$ (Collins, 1999; Charniak, 2000). Generative parsing models are very convenient because training consists of computing probability estimates from counts of parsing events in the training set. However, generative models make complicated and poorly justified independence assumptions and estimations, so we might expect better performance from discriminatively trained models, as has been shown for other tasks like document classification (Joachims, 2002) and shallow parsing (Sha and Pereira, 2003). Ratnaparkhi's conditional maximum entropy model (Ratnaparkhi, 1999), trained to maximize conditional likelihood $P(\boldsymbol{y}|\boldsymbol{x})$ of the training data, performed nearly as well as generative models of the same vintage even though it scores parsing decisions in isolation and thus may suffer from the label bias problem (Lafferty et al., 2001).

Discriminatively trained parsers that score entire trees for a given sentence have only recently been investigated (Riezler et al., 2002; Clark and Curran, 2004; Collins and Roark, 2004; Taskar et al., 2004). The most likely reason for this is that discriminative training requires repeatedly reparsing the training corpus with the current model to determine the parameter updates that will improve the training criterion. The reparsing cost is already quite high for simple context-free models with $O(n^3)$ parsing complexity, but it becomes prohibitive for lexicalized grammars with $O(n^5)$ parsing complexity.

Dependency trees are an alternative syntactic representation with a long history (Hudson, 1984). Dependency trees capture important aspects of functional relationships between words and have been shown to be useful in many applications including relation extraction (Culotta and Sorensen, 2004), paraphrase acquisition (Shinyama et al., 2002) and machine translation (Ding and Palmer, 2005). Yet, they can be parsed in $O(n^3)$ time (Eisner, 1996). Therefore, dependency parsing is a potential "sweet spot" that deserves investigation. We focus here on *projective* dependency trees in which a word is the parent of all of its arguments, and dependencies are non-crossing with respect to word order (see Figure 1). However, there are cases where crossing dependencies may occur, as is the case for Czech (Hajič, 1998). Edges in a dependency tree may be typed (for instance to indicate grammatical function). Though we focus on the simpler non-typed

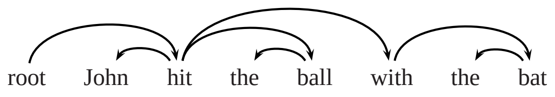


Figure 1: An example dependency tree.

case, all algorithms are easily extendible to typed structures.

The following work on dependency parsing is most relevant to our research. Eisner (1996) gave a generative model with a cubic parsing algorithm based on an edge factorization of trees. Yamada and Matsumoto (2003) trained support vector machines (SVM) to make parsing decisions in a shift-reduce dependency parser. As in Ratnaparkhi's parser, the classifiers are trained on individual decisions rather than on the overall quality of the parse. Nivre and Scholz (2004) developed a history-based learning model. Their parser uses a hybrid bottom-up/top-down linear-time heuristic parser and the ability to label edges with semantic types. The accuracy of their parser is lower than that of Yamada and Matsumoto (2003).

We present a new approach to training dependency parsers, based on the online large-margin learning algorithms of Crammer and Singer (2003) and Crammer et al. (2003). Unlike the SVM parser of Yamada and Matsumoto (2003) and Ratnaparkhi's parser, our parsers are trained to maximize the accuracy of the overall tree.

Our approach is related to those of Collins and Roark (2004) and Taskar et al. (2004) for phrase structure parsing. Collins and Roark (2004) presented a linear parsing model trained with an averaged perceptron algorithm. However, to use parse features with sufficient history, their parsing algorithm must prune heuristically most of the possible parses. Taskar et al. (2004) formulate the parsing problem in the large-margin structured classification setting (Taskar et al., 2003), but are limited to parsing sentences of 15 words or less due to computation time. Though these approaches represent good first steps towards discriminatively-trained parsers, they have not yet been able to display the benefits of discriminative training that have been seen in named-entity extraction and shallow parsing.

Besides simplicity, our method is efficient and accurate, as we demonstrate experimentally on English and Czech treebank data.

2 System Description

2.1 Definitions and Background

In what follows, the generic sentence is denoted by $\boldsymbol{x}$ (possibly subscripted); the ith word of $\boldsymbol{x}$ is denoted by x_i. The generic dependency tree is denoted by $\boldsymbol{y}$. If $\boldsymbol{y}$ is a dependency tree for sentence $\boldsymbol{x}$, we write $(i, j) \in \boldsymbol{y}$ to indicate that there is a directed edge from word x_i to word x_j in the tree, that is, x_i is the parent of x_j. $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{T}$ denotes the training data.

We follow the edge based factorization method of Eisner (1996) and define the score of a dependency tree as the sum of the score of all edges in the tree,

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j)\in\boldsymbol{y}} s(i, j) = \sum_{(i,j)\in\boldsymbol{y}} \mathbf{w} \cdot \mathbf{f}(i, j)$$

where $\mathbf{f}(i, j)$ is a high-dimensional binary feature representation of the edge from x_i to x_j. For example, in the dependency tree of Figure 1, the following feature would have a value of 1:

$$f(i,j) = \begin{cases} 1 \text{ if } x_i\text{='hit' and } x_j\text{='ball'} \\ 0 \text{ otherwise.} \end{cases}$$

In general, any real-valued feature may be used, but we use binary features for simplicity. The feature weights in the weight vector $\mathbf{w}$ are the parameters that will be learned during training. Our training algorithms are iterative. We denote by $\mathbf{w}^{(i)}$ the weight vector after the i^{th} training iteration.

Finally we define $\mathrm{dt}(\boldsymbol{x})$ as the set of possible dependency trees for the input sentence $\boldsymbol{x}$ and $\mathrm{best}_k(\boldsymbol{x}; \mathbf{w})$ as the set of k dependency trees in $\mathrm{dt}(\boldsymbol{x})$ that are given the highest scores by weight vector $\mathbf{w}$, with ties resolved by an arbitrary but fixed rule.

Three basic questions must be answered for models of this form: how to find the dependency tree $\boldsymbol{y}$ with highest score for sentence $\boldsymbol{x}$; how to learn an appropriate weight vector $\mathbf{w}$ from the training data; and finally, what feature representation $\mathbf{f}(i, j)$ should be used. The following sections address each of these questions.

2.2 Parsing Algorithm

Given a feature representation for edges and a weight vector $\mathbf{w}$, we seek the dependency tree or

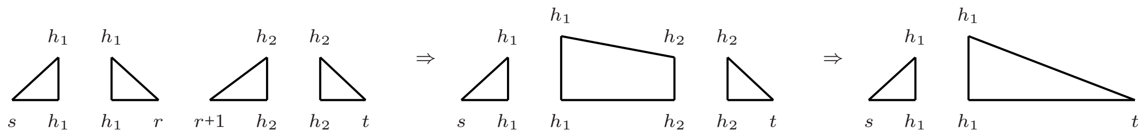


Figure 2: $O(n^3)$ algorithm of Eisner (1996), needs to keep 3 indices at any given stage.

trees that maximize the score function, $s(\boldsymbol{x}, \boldsymbol{y})$. The primary difficulty is that for a given sentence of length n there are exponentially many possible dependency trees. Using a slightly modified version of a lexicalized CKY chart parsing algorithm, it is possible to generate and represent these sentences in a forest that is $O(n^5)$ in size and takes $O(n^5)$ time to create.

Eisner (1996) made the observation that if the head of each chart item is on the left or right periphery, then it is possible to parse in $O(n^3)$. The idea is to parse the left and right dependents of a word independently and combine them at a later stage. This removes the need for the additional head indices of the $O(n^5)$ algorithm and requires only two additional binary variables that specify the direction of the item (either gathering left dependents or gathering right dependents) and whether an item is complete (available to gather more dependents). Figure 2 shows the algorithm schematically. As with normal CKY parsing, larger elements are created bottom-up from pairs of smaller elements.

Eisner showed that his algorithm is sufficient for both searching the space of dependency parses and, with slight modification, finding the highest scoring tree $\boldsymbol{y}$ for a given sentence $\boldsymbol{x}$ under the edge factorization assumption. Eisner and Satta (1999) give a cubic algorithm for lexicalized phrase structures. However, it only works for a limited class of languages in which tree spines are regular. Furthermore, there is a large grammar constant, which is typically in the thousands for treebank parsers.

2.3 Online Learning

Figure 3 gives pseudo-code for the generic online learning setting. A single training instance is considered on each iteration, and parameters updated by applying an algorithm-specific update rule to the instance under consideration. The algorithm in Figure 3 returns an *averaged* weight vector: an auxiliary weight vector **v** is maintained that accumulates

```
Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{T}$
1. $\mathbf{w}_0 = 0;\ \mathbf{v} = 0;\ i = 0$
2. for $n : 1..N$
3.     for $t : 1..T$
4.         $\mathbf{w}^{(i+1)}$ = update $\mathbf{w}^{(i)}$ according to instance $(\boldsymbol{x}_t, \boldsymbol{y}_t)$
5.         $\mathbf{v} = \mathbf{v} + \mathbf{w}^{(i+1)}$
6.         $i = i + 1$
7. $\mathbf{w} = \mathbf{v}/(N * T)$
```

Figure 3: Generic online learning algorithm.

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 (Collins, 2002).

2.3.1 MIRA

Crammer and Singer (2001) developed a natural method for large-margin multi-class classification, which was later extended by Taskar et al. (2003) to structured classification:

$$\begin{aligned}&\min \|\mathbf{w}\|\\ &\text{s.t.}\quad s(\boldsymbol{x},\boldsymbol{y}) - s(\boldsymbol{x},\boldsymbol{y}') \geq L(\boldsymbol{y},\boldsymbol{y}')\\ &\forall(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{T},\ \boldsymbol{y}' \in \text{dt}(\boldsymbol{x})\end{aligned}$$

where $L(\boldsymbol{y}, \boldsymbol{y}')$ is a real-valued loss for the tree $\boldsymbol{y}'$ relative to the correct tree $\boldsymbol{y}$. We define the loss of a dependency tree as the number of words that have the incorrect parent. Thus, the largest loss a dependency tree can have is the length of the sentence.

Informally, this update looks to create a margin between the correct dependency tree and each incorrect dependency tree at least as large as the loss of the incorrect tree. The more errors a tree has, the farther away its score will be from the score of the correct tree. In order to avoid a blow-up in the norm of the weight vector we minimize it subject to constraints that enforce the desired margin between the correct and incorrect trees[1].

[1]The constraints may be unsatisfiable, in which case we can relax them with slack variables as in SVM training.

The Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003; Crammer et al., 2003) employs this optimization directly within the online framework. On each update, MIRA 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. This can be formalized by substituting the following update into line 4 of the generic online algorithm,

$$\begin{aligned}&\min \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\| \\ &\text{s.t. } s(\boldsymbol{x}_t, \boldsymbol{y}_t) - s(\boldsymbol{x}_t, \boldsymbol{y}') \geq L(\boldsymbol{y}_t, \boldsymbol{y}') \\ &\forall \boldsymbol{y}' \in \text{dt}(\boldsymbol{x}_t)\end{aligned} \tag{1}$$

This is a standard quadratic programming problem that can be easily solved using Hildreth's algorithm (Censor and Zenios, 1997). Crammer and Singer (2003) and Crammer et al. (2003) provide an analysis of both the online generalization error and convergence properties of MIRA. In equation (1), $s(\boldsymbol{x}, \boldsymbol{y})$ is calculated with respect to the weight vector after optimization, $\mathbf{w}^{(i+1)}$.

To apply MIRA to dependency parsing, we can simply see parsing as a multi-class classification problem in which each dependency tree is one of many possible classes for a sentence. However, that interpretation fails computationally because a general sentence has exponentially many possible dependency trees and thus exponentially many margin constraints.

To circumvent this problem we make the assumption that the constraints that matter for large margin optimization are those involving the incorrect trees $\boldsymbol{y}'$ with the highest scores $s(\boldsymbol{x}, \boldsymbol{y}')$. The resulting optimization made by MIRA (see Figure 3, line 4) would then be:

$$\begin{aligned}&\min \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\| \\ &\text{s.t.} \quad s(\boldsymbol{x}_t, \boldsymbol{y}_t) - s(\boldsymbol{x}_t, \boldsymbol{y}') \geq L(\boldsymbol{y}_t, \boldsymbol{y}') \\ &\forall \boldsymbol{y}' \in \text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)})\end{aligned}$$

reducing the number of constraints to the constant k. We tested various values of k on a development data set and found that small values of k are sufficient to achieve close to best performance, justifying our assumption. In fact, as k grew we began to observe a slight degradation of performance, indicating some overfitting to the training data. All the experiments presented here use $k = 5$. The Eisner (1996) algorithm can be modified to find the k-best trees while only adding an additional $O(k \log k)$ factor to the runtime (Huang and Chiang, 2005).

A more common approach is to factor the structure of the output space to yield a polynomial set of local constraints (Taskar et al., 2003; Taskar et al., 2004). One such factorization for dependency trees is

$$\begin{aligned}&\min \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\| \\ &\text{s.t. } s(l, j) - s(k, j) \geq 1 \\ &\forall (l, j) \in \boldsymbol{y}_t, (k, j) \notin \boldsymbol{y}_t\end{aligned}$$

It is trivial to show that if these $O(n^2)$ constraints are satisfied, then so are those in (1). We implemented this model, but found that the required training time was much larger than the k-best formulation and typically did not improve performance. Furthermore, the k-best formulation is more flexible with respect to the loss function since it does not assume the loss function can be factored into a sum of terms for each dependency.

2.4 Feature Set

Finally, we need a suitable feature representation $\mathbf{f}(i, j)$ for each dependency. The basic features in our model are outlined in Table 1a and b. All features are conjoined with the direction of attachment as well as the distance between the two words being attached. These features represent a system of back-off from very specific features over words and part-of-speech tags to less sparse features over just part-of-speech tags. These features are added for both the entire words as well as the 5-gram prefix if the word is longer than 5 characters.

Using just features over the parent-child node pairs in the tree was not enough for high accuracy, because all attachment decisions were made outside of the context in which the words occurred. To solve this problem, we added two other types of features, which can be seen in Table 1c. Features of the first type look at words that occur between a child and its parent. These features 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, since a)

Basic Uni-gram Features
p-word, p-pos
p-word
p-pos
c-word, c-pos
c-word
c-pos

b)

Basic Big-ram 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

c)

In Between POS Features
p-pos, b-pos, c-pos
Surrounding Word POS Features
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 1: Features used by system. 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.

it would typically rule out situations when a noun attached to another noun with a verb in between, which is a very uncommon phenomenon.

The second type of feature provides the local context of the attachment, that is, the words before and after the parent-child pair. This feature took 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 to various trigrams where one of the local context POS tags was removed. Adding these two features resulted in a large improvement in performance and brought the system to state-of-the-art accuracy.

2.5 System Summary

Besides performance (see Section 3), the approach to dependency parsing we described has several other advantages. The system is very general and contains no language specific enhancements. In fact, the results we report for English and Czech use identical features, though are obviously trained on different data. The online learning algorithms themselves are intuitive and easy to implement.

The efficient $O(n^3)$ parsing algorithm of Eisner allows the system to search the entire space of dependency trees while parsing thousands of sentences in a few minutes, which is crucial for discriminative training. We compare the speed of our model to a standard lexicalized phrase structure parser in Section 3.1 and show a significant improvement in parsing times on the testing data.

The major limiting factor of the system is its restriction to features over single dependency attachments. Often, when determining the next dependent for a word, it would be useful to know previous attachment decisions and incorporate these into the features. It is fairly straightforward to modify the parsing algorithm to store previous attachments. However, any modification would result in an asymptotic increase in parsing complexity.

3 Experiments

We tested our methods experimentally on the English Penn Treebank (Marcus et al., 1993) and on the Czech Prague Dependency Treebank (Hajič, 1998). All experiments were run on a dual 64-bit AMD Opteron 2.4GHz processor.

To create dependency structures from the Penn Treebank, we used the extraction rules of Yamada and Matsumoto (2003), which are an approximation to the lexicalization rules of Collins (1999). We split the data into three parts: sections 02-21 for training, section 22 for development and section 23 for evaluation. Currently the system has $6,998,447$ features. Each instance only uses a tiny fraction of these features making sparse vector calculations possible. Our system assumes POS tags as input and uses the tagger of Ratnaparkhi (1996) to provide tags for the development and evaluation sets.

Table 2 shows the performance of the systems that were compared. Y&M2003 is the SVM-shift-reduce parsing model of Yamada and Matsumoto (2003), N&S2004 is the memory-based learner of Nivre and Scholz (2004) and MIRA is the the system we have described. We also implemented an averaged perceptron system (Collins, 2002) (another online learning algorithm) for comparison. This table compares only *pure* dependency parsers that do

	English			Czech		
	Accuracy	**Root**	**Complete**	**Accuracy**	**Root**	**Complete**
Y&M2003	90.3	91.6	38.4	-	-	-
N&S2004	87.3	84.3	30.4	-	-	-
Avg. Perceptron	90.6	94.0	36.5	82.9	88.0	30.3
MIRA	90.9	94.2	37.5	83.3	88.6	31.3

Table 2: Dependency parsing results for English and Czech. *Accuracy* is the number of words that correctly identified their parent in the tree. *Root* is the number of trees in which the root word was correctly identified. For Czech this is f-measure since a sentence may have multiple roots. *Complete* is the number of sentences for which the entire dependency tree was correct.

not exploit phrase structure. We ensured that the gold standard dependencies of all systems compared were identical.

Table 2 shows that the model described here performs as well or better than previous comparable systems, including that of Yamada and Matsumoto (2003). Their method has the potential advantage that SVM batch training takes into account all of the constraints from all training instances in the optimization, whereas online training only considers constraints from one instance at a time. However, they are fundamentally limited by their approximate search algorithm. In contrast, our system searches the entire space of dependency trees and most likely benefits greatly from this. This difference is amplified when looking at the percentage of trees that correctly identify the root word. The models that search the entire space will not suffer from bad approximations made early in the search and thus are more likely to identify the correct root, whereas the approximate algorithms are prone to error propagation, which culminates with attachment decisions at the top of the tree. When comparing the two online learning models, it can be seen that MIRA outperforms the averaged perceptron method. This difference is statistically significant, $p < 0.005$ (McNemar test on head selection accuracy).

In our Czech experiments, we used the dependency trees annotated in the Prague Treebank, and the predefined training, development and evaluation sections of this data. The number of sentences in this data set is nearly twice that of the English treebank, leading to a very large number of features — $13,450,672$. But again, each instance uses just a handful of these features. For POS tags we used the automatically generated tags in the data set. Though we made no language specific model changes, we did need to make some data specific changes. In particular, we used the method of Collins et al. (1999) to simplify part-of-speech tags since the rich tags used by Czech would have led to a large but rarely seen set of POS features.

The model based on MIRA also performs well on Czech, again slightly outperforming averaged perceptron. Unfortunately, we do not know of any other parsing systems tested on the same data set. The Czech parser of Collins et al. (1999) was run on a different data set and most other dependency parsers are evaluated using English. Learning a model from the Czech training data is somewhat problematic since it contains some crossing dependencies which cannot be parsed by the Eisner algorithm. One trick is to rearrange the words in the training set so that all trees are nested. This at least allows the training algorithm to obtain reasonably low error on the training set. We found that this did improve performance slightly to 83.6% accuracy.

3.1 Lexicalized Phrase Structure Parsers

It is well known that dependency trees extracted from lexicalized phrase structure parsers (Collins, 1999; Charniak, 2000) typically are more accurate than those produced by pure dependency parsers (Yamada and Matsumoto, 2003). We compared our system to the Bikel re-implementation of the Collins parser (Bikel, 2004; Collins, 1999) trained with the same head rules of our system. There are two ways to extract dependencies from lexicalized phrase structure. The first is to use the automatically generated dependencies that are explicit in the lexicalization of the trees, we call this system *Collins-auto*. The second is to take just the phrase structure output of the parser and run the automatic head rules over it to extract the dependencies, we call this sys-

	English				
	Accuracy	Root	Complete	Complexity	Time
Collins-auto	88.2	92.3	36.1	$O(n^5)$	98m 21s
Collins-rules	91.4	95.1	42.6	$O(n^5)$	98m 21s
MIRA-Normal	90.9	94.2	37.5	$O(n^3)$	5m 52s
MIRA-Collins	92.2	95.8	42.9	$O(n^5)$	105m 08s

Table 3: Results comparing our system to those based on the Collins parser. *Complexity* represents the computational complexity of each parser and *Time* the CPU time to parse sec. 23 of the Penn Treebank.

tem *Collins-rules*. Table 3 shows the results comparing our system, *MIRA-Normal*, to the Collins parser for English. All systems are implemented in Java and run on the same machine.

Interestingly, the dependencies that are automatically produced by the Collins parser are worse than those extracted statically using the head rules. Arguably, this displays the artificialness of English dependency parsing using dependencies automatically extracted from treebank phrase-structure trees. Our system falls in-between, better than the automatically generated dependency trees and worse than the head-rule extracted trees.

Since the dependencies returned from our system are better than those actually learnt by the Collins parser, one could argue that our model is actually learning to parse dependencies more accurately. However, phrase structure parsers are built to maximize the accuracy of the phrase structure and use lexicalization as just an additional source of information. Thus it is not too surprising that the dependencies output by the Collins parser are not as accurate as our system, which is trained and built to maximize accuracy on dependency trees. In complexity and run-time, our system is a huge improvement over the Collins parser.

The final system in Table 3 takes the output of *Collins-rules* and adds a feature to *MIRA-Normal* that indicates for given edge, whether the Collins parser believed this dependency actually exists, we call this system *MIRA-Collins*. This is a well known discriminative training trick — using the suggestions of a generative system to influence decisions. This system can essentially be considered a corrector of the Collins parser and represents a significant improvement over it. However, there is an added complexity with such a model as it requires the output of the $O(n^5)$ Collins parser.

	k=1	k=2	k=5	k=10	k=20
Accuracy	90.73	90.82	90.88	90.92	90.91
Train Time	183m	235m	627m	1372m	2491m

Table 4: Evaluation of k-best MIRA approximation.

3.2 k-best MIRA Approximation

One question that can be asked is how justifiable is the k-best MIRA approximation. Table 4 indicates the accuracy on testing and the time it took to train models with $k = 1, 2, 5, 10, 20$ for the English data set. Even though the parsing algorithm is proportional to $O(k \log k)$, empirically, the training times scale linearly with k. Peak performance is achieved very early with a slight degradation around k=20. The most likely reason for this phenomenon is that the model is overfitting by ensuring that even unlikely trees are separated from the correct tree proportional to their loss.

4 Summary

We described a successful new method for training dependency parsers. We use simple linear parsing models trained with margin-sensitive online training algorithms, achieving state-of-the-art performance with relatively modest training times and no need for pruning heuristics. We evaluated the system on both English and Czech data to display state-of-the-art performance without any language specific enhancements. Furthermore, the model can be augmented to include features over lexicalized phrase structure parsing decisions to increase dependency accuracy over those parsers.

We plan on extending our parser in two ways. First, we would add labels to dependencies to represent grammatical roles. Those labels are very important for using parser output in tasks like information extraction or machine translation. Second, we are looking at model extensions to allow non-projective dependencies, which occur in languages such as Czech, German and Dutch.

Acknowledgments: We thank Jan Hajič for answering queries on the Prague treebank, and Joakim Nivre for providing the Yamada and Matsumoto (2003) head rules for English that allowed for a direct comparison with our systems. This work was supported by NSF ITR grants 0205456, 0205448, and 0428193.

References

D.M. Bikel. 2004. Intricacies of Collins parsing model. *Computational Linguistics*.

Y. Censor and S.A. Zenios. 1997. *Parallel optimization : theory, algorithms, and applications*. Oxford University Press.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. NAACL*.

S. Clark and J.R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proc. ACL*.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. ACL*.

M. Collins, J. Hajič, L. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proc. ACL*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.

K. Crammer and Y. Singer. 2001. On the algorithmic implementation of multiclass kernel based vector machines. *JMLR*.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *JMLR*.

K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. 2003. Online passive aggressive algorithms. In *Proc. NIPS*.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. ACL*.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. ACL*.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proc. ACL*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. COLING*.

J. Hajič. 1998. Building a syntactically annotated corpus: The Prague dependency treebank. *Issues of Valency and Meaning*.

L. Huang and D. Chiang. 2005. Better k-best parsing. Technical Report MS-CIS-05-08, University of Pennsylvania.

Richard Hudson. 1984. *Word Grammar*. Blackwell.

T. Joachims. 2002. *Learning to Classify Text using Support Vector Machines*. Kluwer.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*.

J. Nivre and M. Scholz. 2004. Deterministic dependency parsing of english text. In *Proc. COLING*.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. EMNLP*.

A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*.

S. Riezler, T. King, R. Kaplan, R. Crouch, J. Maxwell, and M. Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proc. ACL*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. HLT-NAACL*.

Y. Shinyama, S. Sekine, K. Sudo, and R. Grishman. 2002. Automatic paraphrase acquisition from news articles. In *Proc. HLT*.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin Markov networks. In *Proc. NIPS*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proc. EMNLP*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. IWPT*.

Pseudo-Projective Dependency Parsing

Joakim Nivre and **Jens Nilsson**
School of Mathematics and Systems Engineering
Växjö University
SE-35195 Växjö, Sweden
{nivre,jni}@msi.vxu.se

Abstract

In order to realize the full potential of dependency-based syntactic parsing, it is desirable to allow non-projective dependency structures. We show how a data-driven deterministic dependency parser, in itself restricted to projective structures, can be combined with graph transformation techniques to produce non-projective structures. Experiments using data from the Prague Dependency Treebank show that the combined system can handle non-projective constructions with a precision sufficient to yield a significant improvement in overall parsing accuracy. This leads to the best reported performance for robust non-projective parsing of Czech.

1 Introduction

It is sometimes claimed that one of the advantages of dependency grammar over approaches based on constituency is that it allows a more adequate treatment of languages with variable word order, where discontinuous syntactic constructions are more common than in languages like English (Mel'čuk, 1988; Covington, 1990). However, this argument is only plausible if the formal framework allows non-projective dependency structures, i.e. structures where a head and its dependents may correspond to a discontinuous constituent. From the point of view of computational implementation this can be problematic, since the inclusion of non-projective structures makes the parsing problem more complex and therefore compromises efficiency and in practice also accuracy and robustness. Thus, most broad-coverage parsers based on dependency grammar have been restricted to projective structures. This is true of the widely used link grammar parser for English (Sleator and Temperley, 1993), which uses a dependency grammar of sorts, the probabilistic dependency parser of Eisner (1996), and more recently proposed deterministic dependency parsers (Yamada and Matsumoto, 2003; Nivre et al., 2004). It is also true of the adaptation of the Collins parser for Czech (Collins et al., 1999) and the finite-state dependency parser for Turkish by Oflazer (2003).

This is in contrast to dependency treebanks, e.g. Prague Dependency Treebank (Hajič et al., 2001b), Danish Dependency Treebank (Kromann, 2003), and the METU Treebank of Turkish (Oflazer et al., 2003), which generally allow annotations with non-projective dependency structures. The fact that projective dependency parsers can never exactly reproduce the analyses found in non-projective treebanks is often neglected because of the relative scarcity of problematic constructions. While the proportion of sentences containing non-projective dependencies is often 15–25%, the total proportion of non-projective arcs is normally only 1–2%. As long as the main evaluation metric is dependency accuracy per word, with state-of-the-art accuracy mostly below 90%, the penalty for not handling non-projective constructions is almost negligible. Still, from a theoretical point of view, projective parsing of non-projective structures has the drawback that it rules out perfect accuracy even as an asymptotic goal.

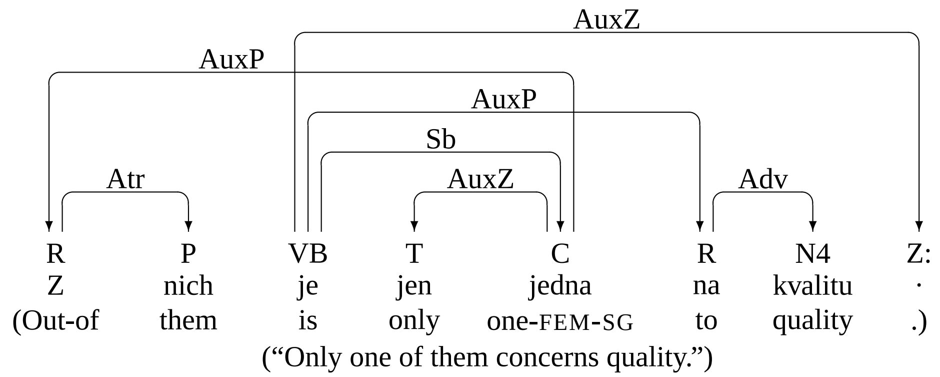


Figure 1: Dependency graph for Czech sentence from the Prague Dependency Treebank[1]

There exist a few robust broad-coverage parsers that produce non-projective dependency structures, notably Tapanainen and Järvinen (1997) and Wang and Harper (2004) for English, Foth et al. (2004) for German, and Holan (2004) for Czech. In addition, there are several approaches to non-projective dependency parsing that are still to be evaluated in the large (Covington, 1990; Kahane et al., 1998; Duchier and Debusmann, 2001; Holan et al., 2001; Hellwig, 2003). Finally, since non-projective constructions often involve long-distance dependencies, the problem is closely related to the recovery of empty categories and non-local dependencies in constituency-based parsing (Johnson, 2002; Dienes and Dubey, 2003; Jijkoun and de Rijke, 2004; Cahill et al., 2004; Levy and Manning, 2004; Campbell, 2004).

In this paper, we show how non-projective dependency parsing can be achieved by combining a data-driven projective parser with special graph transformation techniques. First, the training data for the parser is projectivized by applying a minimal number of lifting operations (Kahane et al., 1998) and encoding information about these lifts in arc labels. When the parser is trained on the transformed data, it will ideally learn not only to construct projective dependency structures but also to assign arc labels that encode information about lifts. By applying an inverse transformation to the output of the parser, arcs with non-standard labels can be lowered to their proper place in the dependency graph, giving rise to non-projective structures. We call this pseudo-projective dependency parsing, since it is based on a notion of pseudo-projectivity (Kahane et al., 1998).

The rest of the paper is structured as follows. In section 2 we introduce the graph transformation techniques used to projectivize and deprojectivize dependency graphs, and in section 3 we describe the data-driven dependency parser that is the core of our system. We then evaluate the approach in two steps. First, in section 4, we evaluate the graph transformation techniques in themselves, with data from the Prague Dependency Treebank and the Danish Dependency Treebank. In section 5, we then evaluate the entire parsing system by training and evaluating on data from the Prague Dependency Treebank.

2 Dependency Graph Transformations

We assume that the goal in dependency parsing is to construct a labeled dependency graph of the kind depicted in Figure 1. Formally, we define dependency graphs as follows:

1. Let $R = \{r_1, \ldots, r_m\}$ be the set of permissible dependency types (arc labels).
2. A dependency graph for a string of words $W = w_1 \cdots w_n$ is a labeled directed graph $D = (W, A)$, where
   (a) W is the set of nodes, i.e. word tokens in the input string, ordered by a linear precedence relation $<$,
   (b) A is a set of labeled arcs (w_i, r, w_j), where $w_i, w_j \in W$, $r \in R$,
   (c) for every $w_j \in W$, there is at most one arc $(w_i, r, w_j) \in A$.

[1]The dependency graph has been modified to make the final period a dependent of the main verb instead of being a dependent of a special root node for the sentence.

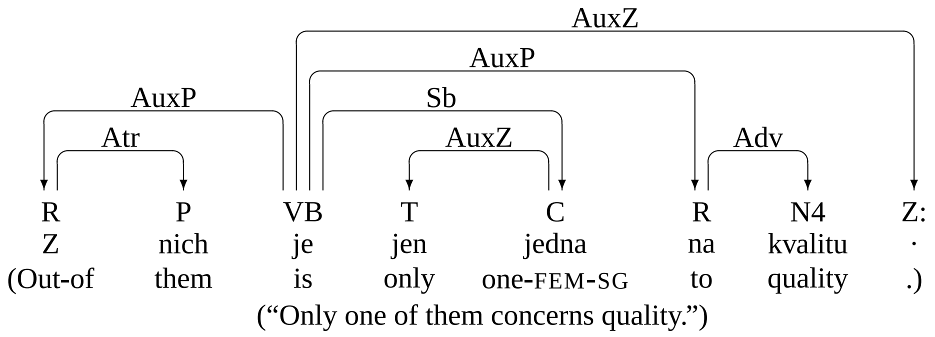


Figure 2: Projectivized dependency graph for Czech sentence

3. A graph $D = (W, A)$ is well-formed iff it is acyclic and connected.

If $(w_i, r, w_j) \in A$, we say that w_i is the head of w_j and w_j a dependent of w_i. In the following, we use the notation $w_i \xrightarrow{r} w_j$ to mean that $(w_i, r, w_j) \in A$; we also use $w_i \rightarrow w_j$ to denote an arc with unspecified label and $w_i \rightarrow^* w_j$ for the reflexive and transitive closure of the (unlabeled) arc relation.

The dependency graph in Figure 1 satisfies all the defining conditions above, but it fails to satisfy the condition of projectivity (Kahane et al., 1998):

1. An arc $w_i \rightarrow w_k$ is projective iff, for every word w_j occurring between w_i and w_k in the string ($w_i < w_j < w_k$ or $w_i > w_j > w_k$), $w_i \rightarrow^* w_j$.

2. A dependency graph $D = (W, A)$ is projective iff every arc in A is projective.

The arc connecting the head *jedna* (one) to the dependent *Z* (out-of) spans the token *je* (is), which is not dominated by *jedna*.

As observed by Kahane et al. (1998), any (non-projective) dependency graph can be transformed into a projective one by a lifting operation, which replaces each non-projective arc $w_j \rightarrow w_k$ by a projective arc $w_i \rightarrow w_k$ such that $w_i \rightarrow^* w_j$ holds in the original graph. Here we use a slightly different notion of lift, applying to individual arcs and moving their head upwards one step at a time:

$$\textsc{Lift}(w_j \rightarrow w_k) \;=\; \begin{cases} w_i \rightarrow w_k \text{ if } w_i \rightarrow w_j \\ \text{undefined otherwise} \end{cases}$$

Intuitively, lifting an arc makes the word w_k dependent on the head w_i of its original head w_j (which is unique in a well-formed dependency graph), unless w_j is a root in which case the operation is undefined (but then $w_j \rightarrow w_k$ is necessarily projective if the dependency graph is well-formed).

Projectivizing a dependency graph by lifting non-projective arcs is a nondeterministic operation in the general case. However, since we want to preserve as much of the original structure as possible, we are interested in finding a transformation that involves a minimal number of lifts. Even this may be nondeterministic, in case the graph contains several non-projective arcs whose lifts interact, but we use the following algorithm to construct a minimal projective transformation $D' = (W, A')$ of a (non-projective) dependency graph $D = (W, A)$:

```
PROJECTIVIZE($W$, $A$)
1  $A' \leftarrow A$
2  **while** $(W, A')$ is non-projective
3      $a \leftarrow$ SMALLEST-NONP-ARC$(A')$
4      $A' \leftarrow (A' - \{a\}) \cup \{$LIFT$(a)\}$
5  **return** $(W, A')$
```

The function SMALLEST-NONP-ARC returns the non-projective arc with the shortest distance from head to dependent (breaking ties from left to right). Applying the function PROJECTIVIZE to the graph in Figure 1 yields the graph in Figure 2, where the problematic arc pointing to *Z* has been lifted from the original head *jedna* to the ancestor *je*. Using the terminology of Kahane et al. (1998), we say that *jedna* is the *syntactic head* of *Z*, while *je* is its *linear head* in the projectivized representation.

Unlike Kahane et al. (1998), we do not regard a projectivized representation as the final target of the parsing process. Instead, we want to apply an in-

	Lifted arc label	**Path labels**	**Number of labels**
Baseline	d	p	n
Head	$d{\uparrow}h$	p	$n(n+1)$
Head+Path	$d{\uparrow}h$	$p{\downarrow}$	$2n(n+1)$
Path	$d{\uparrow}$	$p{\downarrow}$	$4n$

Table 1: Encoding schemes (d = dependent, h = syntactic head, p = path; n = number of dependency types)

verse transformation to recover the underlying (non-projective) dependency graph. In order to facilitate this task, we extend the set of arc labels to encode information about lifting operations. In principle, it would be possible to encode the exact position of the syntactic head in the label of the arc from the linear head, but this would give a potentially infinite set of arc labels and would make the training of the parser very hard. In practice, we can therefore expect a trade-off such that increasing the amount of information encoded in arc labels will cause an increase in the accuracy of the inverse transformation but a decrease in the accuracy with which the parser can construct the labeled representations. To explore this tradeoff, we have performed experiments with three different encoding schemes (plus a baseline), which are described schematically in Table 1.

The baseline simply retains the original labels for all arcs, regardless of whether they have been lifted or not, and the number of distinct labels is therefore simply the number n of distinct dependency types.[2] In the first encoding scheme, called **Head**, we use a new label $d{\uparrow}h$ for each lifted arc, where d is the dependency relation between the syntactic head and the dependent in the non-projective representation, and h is the dependency relation that the syntactic head has to its own head in the underlying structure. Using this encoding scheme, the arc from *je* to *Z* in Figure 2 would be assigned the label AuxP↑Sb (signifying an AuxP that has been lifted from a Sb). In the second scheme, **Head+Path**, we in addition modify the label of every arc along the lifting path from the syntactic to the linear head so that if the original label is p the new label is $p{\downarrow}$. Thus, the arc from *je* to *jedna* will be labeled $Sb{\downarrow}$ (to indicate that there is a syntactic head below it). In the third and final scheme, denoted **Path**, we keep the extra information on path labels but drop the information about the syntactic head of the lifted arc, using the label $d{\uparrow}$ instead of $d{\uparrow}h$ (AuxP↑ instead of AuxP↑Sb).

As can be seen from the last column in Table 1, both **Head** and **Head+Path** may theoretically lead to a quadratic increase in the number of distinct arc labels (**Head+Path** being worse than **Head** only by a constant factor), while the increase is only linear in the case of **Path**. On the other hand, we can expect **Head+Path** to be the most useful representation for reconstructing the underlying non-projective dependency graph. In approaching this problem, a variety of different methods are conceivable, including a more or less sophisticated use of machine learning. In the present study, we limit ourselves to an algorithmic approach, using a deterministic breadth-first search. The details of the transformation procedure are slightly different depending on the encoding schemes:

- **Head:** For every arc of the form $w_i \xrightarrow{d\uparrow h} w_n$, we search the graph top-down, left-to-right, breadth-first starting at the head node w_i. If we find an arc $w_l \xrightarrow{h} w_m$, called a *target arc*, we replace $w_i \xrightarrow{d\uparrow h} w_n$ by $w_m \xrightarrow{d} w_n$; otherwise we replace $w_i \xrightarrow{d\uparrow h} w_n$ by $w_i \xrightarrow{d} w_n$ (i.e. we let the linear head be the syntactic head).
- **Head+Path:** Same as **Head**, but the search only follows arcs of the form $w_j \xrightarrow{p\downarrow} w_k$ and a target arc must have the form $w_l \xrightarrow{h\downarrow} w_m$; if no target arc is found, **Head** is used as backoff.
- **Path:** Same as **Head+Path**, but a target arc must have the form $w_l \xrightarrow{p\downarrow} w_m$ and no outgoing arcs of the form $w_m \xrightarrow{p'\downarrow} w_o$; no backoff.

In section 4 we evaluate these transformations with respect to projectivized dependency treebanks, and in section 5 they are applied to parser output. Before

[2]Note that this is a baseline for the parsing experiment only (Experiment 2). For Experiment 1 it is meaningless as a baseline, since it would result in 0% accuracy.

Feature type		Top−1	Top	Next	Next+1	Next+2	Next+3
Word form		+	+	+	+		
Part-of-speech		+	+	+	+	+	+
Dep type of	head		+				
	leftmost dep		+	+			
	rightmost dep		+				

Table 2: Features used in predicting the next parser action

we turn to the evaluation, however, we need to introduce the data-driven dependency parser used in the latter experiments.

3 Memory-Based Dependency Parsing

In the experiments below, we employ a data-driven deterministic dependency parser producing labeled projective dependency graphs,[3] previously tested on Swedish (Nivre et al., 2004) and English (Nivre and Scholz, 2004). The parser builds dependency graphs by traversing the input from left to right, using a stack to store tokens that are not yet complete with respect to their dependents. At each point during the derivation, the parser has a choice between pushing the next input token onto the stack – with or without adding an arc from the token on top of the stack to the token pushed – and popping a token from the stack – with or without adding an arc from the next input token to the token popped. More details on the parsing algorithm can be found in Nivre (2003).

The choice between different actions is in general nondeterministic, and the parser relies on a memory-based classifier, trained on treebank data, to predict the next action based on features of the current parser configuration. Table 2 shows the features used in the current version of the parser. At each point during the derivation, the prediction is based on six word tokens, the two topmost tokens on the stack, and the next four input tokens. For each token, three types of features may be taken into account: the word form; the part-of-speech assigned by an automatic tagger; and labels on previously assigned dependency arcs involving the token – the arc from its head and the arcs to its leftmost and rightmost dependent, respectively. Except for the leftmost dependent of the next input token, dependency type features are limited to tokens on the stack.

The prediction based on these features is a k-nearest neighbor classification, using the IB1 algorithm and $k = 5$, the modified value difference metric (MVDM) and class voting with inverse distance weighting, as implemented in the TiMBL software package (Daelemans et al., 2003). More details on the memory-based prediction can be found in Nivre et al. (2004) and Nivre and Scholz (2004).

4 Experiment 1: Treebank Transformation

The first experiment uses data from two dependency treebanks. The Prague Dependency Treebank (PDT) consists of more than 1M words of newspaper text, annotated on three levels, the morphological, analytical and tectogrammatical levels (Hajič, 1998). Our experiments all concern the analytical annotation, and the first experiment is based only on the training part. The Danish Dependency Treebank (DDT) comprises about 100K words of text selected from the Danish PAROLE corpus, with annotation of primary and secondary dependencies (Kromann, 2003). The entire treebank is used in the experiment, but only primary dependencies are considered.[4] In all experiments, punctuation tokens are included in the data but omitted in evaluation scores.

In the first part of the experiment, dependency graphs from the treebanks were projectivized using the algorithm described in section 2. As shown in Table 3, the proportion of sentences containing some non-projective dependency ranges from about 15% in DDT to almost 25% in PDT. However, the overall percentage of non-projective arcs is less than 2% in PDT and less than 1% in DDT. The last four

[3]The graphs satisfy all the well-formedness conditions given in section 2 except (possibly) connectedness. For robustness reasons, the parser may output a set of dependency trees instead of a single tree.

[4]If secondary dependencies had been included, the dependency graphs would not have satisfied the well-formedness conditions formulated in section 2.

					# Lifts in projectivization			
Data set	**# Sentences**	**% NonP**	**# Tokens**	**% NonP**	**1**	**2**	**3**	**>3**
PDT training	73,088	23.15	1,255,333	1.81	93.79	5.60	0.51	0.11
DDT total	5,512	15.48	100,238	0.94	79.49	13.28	4.36	2.87

Table 3: Non-projective sentences and arcs in PDT and DDT (NonP = non-projective)

Data set	**Head**		**H+P**		**Path**	
PDT training (28 labels)	92.3	(230)	99.3	(314)	97.3	(84)
DDT total (54 labels)	92.3	(123)	99.8	(147)	98.3	(99)

Table 4: Percentage of non-projective arcs recovered correctly (number of labels in parentheses)

columns in Table 3 show the distribution of non-projective arcs with respect to the number of lifts required. It is worth noting that, although non-projective constructions are less frequent in DDT than in PDT, they seem to be more deeply nested, since only about 80% can be projectivized with a single lift, while almost 95% of the non-projective arcs in PDT only require a single lift.

In the second part of the experiment, we applied the inverse transformation based on breadth-first search under the three different encoding schemes. The results are given in Table 4. As expected, the most informative encoding, **Head+Path**, gives the highest accuracy with over 99% of all non-projective arcs being recovered correctly in both data sets. However, it can be noted that the results for the least informative encoding, **Path**, are almost comparable, while the third encoding, **Head**, gives substantially worse results for both data sets. We also see that the increase in the size of the label sets for **Head** and **Head+Path** is far below the theoretical upper bounds given in Table 1. The increase is generally higher for PDT than for DDT, which indicates a greater diversity in non-projective constructions.

5 Experiment 2: Memory-Based Parsing

The second experiment is limited to data from PDT.[5] The training part of the treebank was projectivized under different encoding schemes and used to train memory-based dependency parsers, which were run on the test part of the treebank, consisting of 7,507 sentences and 125,713 tokens.[6] The inverse transformation was applied to the output of the parsers and the result compared to the gold standard test set.

Table 5 shows the overall parsing accuracy attained with the three different encoding schemes, compared to the baseline (no special arc labels) and to training directly on non-projective dependency graphs. Evaluation metrics used are Attachment Score (AS), i.e. the proportion of tokens that are attached to the correct head, and Exact Match (EM), i.e. the proportion of sentences for which the dependency graph exactly matches the gold standard. In the labeled version of these metrics (L) both heads and arc labels must be correct, while the unlabeled version (U) only considers heads.

The first thing to note is that projectivizing helps in itself, even if no encoding is used, as seen from the fact that the projective baseline outperforms the non-projective training condition by more than half a percentage point on attachment score, although the gain is much smaller with respect to exact match. The second main result is that the pseudo-projective approach to parsing (using special arc labels to guide an inverse transformation) gives a further improvement of about one percentage point on attachment score. With respect to exact match, the improvement is even more noticeable, which shows quite clearly that even if non-projective dependencies are rare on the token level, they are nevertheless important for getting the global syntactic structure correct.

All improvements over the baseline are statistically significant beyond the 0.01 level (McNemar's

[5]Preliminary experiments using data from DDT indicated that the limited size of the treebank creates a severe sparse data problem with respect to non-projective constructions.

[6]The part-of-speech tagging used in both training and testing was the uncorrected output of an HMM tagger distributed with the treebank; cf. Hajič et al. (2001a).

Encoding	**UAS**	**LAS**	**UEM**	**LEM**
Non-projective	78.5	71.3	28.9	20.6
Baseline	79.1	72.0	29.2	20.7
Head	80.1	72.8	31.6	22.2
Head+Path	80.0	72.8	31.8	22.4
Path	80.0	72.7	31.6	22.0

Table 5: Parsing accuracy (AS = attachment score, EM = exact match; U = unlabeled, L = labeled)

	Unlabeled			**Labeled**		
Encoding	**P**	**R**	**F**	**P**	**R**	**F**
Head	61.3	54.1	57.5	55.2	49.8	52.4
Head+Path	63.9	54.9	59.0	57.9	50.6	54.0
Path	58.2	49.5	53.4	51.0	45.7	48.2

Table 6: Precision, recall and F-measure for non-projective arcs

test). By contrast, when we turn to a comparison of the three encoding schemes it is hard to find any significant differences, and the overall impression is that it makes little or no difference which encoding scheme is used, as long as there is some indication of which words are assigned their linear head instead of their syntactic head by the projective parser. This may seem surprising, given the experiments reported in section 4, but the explanation is probably that the non-projective dependencies that can be recovered at all are of the simple kind that only requires a single lift, where the encoding of path information is often redundant. It is likely that the more complex cases, where path information could make a difference, are beyond the reach of the parser in most cases.

However, if we consider precision, recall and F-measure on non-projective dependencies only, as shown in Table 6, some differences begin to emerge. The most informative scheme, **Head+Path**, gives the highest scores, although with respect to **Head** the difference is not statistically significant, while the least informative scheme, **Path** – with almost the same performance on treebank transformation – is significantly lower ($p < 0.01$). On the other hand, given that all schemes have similar parsing accuracy overall, this means that the **Path** scheme is the least likely to introduce errors on projective arcs.

The overall parsing accuracy obtained with the pseudo-projective approach is still lower than for the best projective parsers. Although the best published results for the Collins parser is 80% UAS (Collins, 1999), this parser reaches 82% when trained on the entire training data set, and an adapted version of Charniak's parser (Charniak, 2000) performs at 84% (Jan Hajič, pers. comm.). However, the accuracy is considerably higher than previously reported results for robust non-projective parsing of Czech, with a best performance of 73% UAS (Holan, 2004).

Compared to related work on the recovery of long-distance dependencies in constituency-based parsing, our approach is similar to that of Dienes and Dubey (2003) in that the processing of non-local dependencies is partly integrated in the parsing process, via an extension of the set of syntactic categories, whereas most other approaches rely on post-processing only. However, while Dienes and Dubey recognize empty categories in a pre-processing step and only let the parser find their antecedents, we use the parser both to detect dislocated dependents and to predict either the type or the location of their syntactic head (or both) and use post-processing only to transform the graph in accordance with the parser's analysis.

6 Conclusion

We have presented a new method for non-projective dependency parsing, based on a combination of data-driven projective dependency parsing and graph transformation techniques. The main result is that the combined system can recover non-projective dependencies with a precision sufficient to give a significant improvement in overall parsing accuracy, especially with respect to the exact match criterion, leading to the best reported performance for robust non-projective parsing of Czech.

Acknowledgements

This work was supported in part by the Swedish Research Council (621-2002-4207). Memory-based classifiers for the experiments were created using TiMBL (Daelemans et al., 2003). Special thanks to Jan Hajič and Matthias Trautner Kromann for assistance with the Czech and Danish data, respectively, and to Jan Hajič, Tomáš Holan, Dan Zeman and three anonymous reviewers for valuable comments on a preliminary version of the paper.

References

Cahill, A., Burke, M., O'Donovan, R., Van Genabith, J. and Way, A. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of ACL*.

Campbell, R. 2004. Using linguistic principles to recover empty categories. In *Proceedings of ACL*.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*.

Collins, M., Hajič, J., Brill, E., Ramshaw, L. and Tillmann, C. 1999. A statistical parser for Czech. In *Proceedings of ACL*.

Collins, M. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Covington, M. A. 1990. Parsing discontinuous constituents in dependency grammar. *Computational Linguistics*, 16:234–236.

Daelemans, W., Zavrel, J., van der Sloot, K. and van den Bosch, A. 2003. TiMBL: Tilburg Memory Based Learner, version 5.0, Reference Guide. Technical Report ILK 03-10, Tilburg University, ILK.

Dienes, P. and Dubey, A. 2003. Deep syntactic processing by combining shallow methods. In *Proceedings of ACL*.

Duchier, D. and Debusmann, R. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of ACL*.

Eisner, J. M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*.

Foth, K., Daum, M. and Menzel, W. 2004. A broad-coverage parser for German based on defeasible constraints. In *Proceedings of KONVENS*.

Hajič, J., Krbec, P., Oliva, K., Kveton, P. and Petkevic, V. 2001. Serial combination of rules and statistics: A case study in Czech tagging. In *Proceedings of ACL*.

Hajič, J., Vidova Hladka, B., Panevová, J., Hajičová, E., Sgall, P. and Pajas, P. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

Hajič, J. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, pages 106–132. Karolinum.

Hellwig, P. 2003. Dependency unification grammar. In *Dependency and Valency*, pages 593–635. Walter de Gruyter.

Holan, T., Kuboň, V. and Plátek, M. 2001. Word-order relaxations and restrictions within a dependency grammar. In *Proceedings of IWPT*.

Holan, T. 2004. Tvorba zavislostniho syntaktickeho analyzatoru. In *Proceedings of MIS'2004*.

Jijkoun, V. and de Rijke, M. 2004. Enriching the output of a parser using memory-based learning. In *Proceedings of ACL*.

Johnson, M. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of ACL*.

Kahane, S., Nasr, A. and Rambow, O. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *Proceedings of ACL-COLING*.

Kromann, M. T. 2003. The Danish Dependency Treebank and the DTAG treebank tool. In *Proceedings of TLT 2003*.

Levy, R. and Manning, C. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of ACL*.

Mel'čuk, I. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Nivre, J. and Scholz, M. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING*.

Nivre, J., Hall, J. and Nilsson, J. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*.

Nivre, J. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*.

Oflazer, K., Say, B., Hakkani-Tür, D. Z. and Tür, G. 2003. Building a Turkish treebank. In *Treebanks: Building and Using Parsed Corpora*, pages 261–277. Kluwer Academic Publishers.

Oflazer, K. 2003. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29:515–544.

Sleator, D. and Temperley, D. 1993. Parsing English with a link grammar. In *Proceedings of IWPT*.

Tapanainen, P. and Järvinen, T. 1997. A non-projective dependency parser. In *Proceedings of ANLP*.

Wang, W. and Harper, M. P. 2004. A statistical constraint dependency grammar (CDG) parser. In *Proceedings of the Workshop in Incremental Parsing (ACL)*.

Yamada, H. and Matsumoto, Y. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*.

The Distributional Inclusion Hypotheses and Lexical Entailment

Maayan Geffet
School of Computer Science and Engineering
Hebrew University, Jerusalem, Israel, 91904
mary@cs.huji.ac.il

Ido Dagan
Department of Computer Science
Bar-Ilan University, Ramat-Gan, Israel, 52900
dagan@cs.biu.ac.il

Abstract

This paper suggests refinements for the Distributional Similarity Hypothesis. Our proposed hypotheses relate the distributional behavior of pairs of words to *lexical entailment* – a tighter notion of semantic similarity that is required by many NLP applications. To automatically explore the validity of the defined hypotheses we developed an *inclusion testing algorithm* for characteristic features of two words, which incorporates corpus and web-based feature sampling to overcome data sparseness. The degree of hypotheses validity was then empirically tested and manually analyzed with respect to the word sense level. In addition, the above testing algorithm was exploited to improve lexical entailment acquisition.

1 Introduction

Distributional Similarity between words has been an active research area for more than a decade. It is based on the general idea of Harris' Distributional Hypothesis, suggesting that words that occur within similar contexts are semantically similar (Harris, 1968). Concrete similarity measures compare a pair of weighted context feature vectors that characterize two words (Church and Hanks, 1990; Ruge, 1992; Pereira et al., 1993; Grefenstette, 1994; Lee, 1997; Lin, 1998; Pantel and Lin, 2002; Weeds and Weir, 2003).

As it turns out, distributional similarity captures a somewhat loose notion of semantic similarity (see Table 1). It does not ensure that the meaning of one word is *preserved* when replacing it with the other one in some context.

However, many semantic information-oriented applications like Question Answering, Information Extraction and Paraphrase Acquisition require a tighter similarity criterion, as was also demonstrated by papers at the recent PASCAL Challenge on Recognizing Textual Entailment (Dagan et al., 2005). In particular, all these applications need to know when the meaning of one word can be inferred (entailed) from another word, so that one word could substitute the other in some contexts. This relation corresponds to several lexical semantic relations, such as synonymy, hyponymy and some cases of meronymy. For example, in Question Answering, the word *company* in a question can be substituted in the text by *firm* (synonym), *automaker* (hyponym) or *division* (meronym). Unfortunately, existing manually constructed resources of lexical semantic relations, such as WordNet, are not exhaustive and comprehensive enough for a variety of domains and thus are not sufficient as a sole resource for application needs[1].

Most works that attempt to learn such concrete lexical semantic relations employ a co-occurrence pattern-based approach (Hearst, 1992; Ravichandran and Hovy, 2002; Moldovan et al., 2004). Typically, they use a set of predefined lexico-syntactic patterns that characterize specific semantic relations. If a candidate word pair (like *company-automaker*) co-occurs within the same sentence satisfying a concrete pattern (like " …companies, *such as* automakers"), then it is expected that the corresponding semantic relation holds between these words (hypernym-hyponym in this example).

In recent work (Geffet and Dagan, 2004) we explored the correspondence between the distributional characterization of two words (which may hardly co-occur, as is usually the case for syno-

[1]We found that less than 20% of the lexical entailment relations extracted by our method appeared as direct or indirect WordNet relations (synonyms, hyponyms or meronyms).

<=> element, component	**<=> gap, spread**	*** town, airport**	**<= loan, mortgage**
=> government, body	*** warplane, bomb**	**<=> program, plan**	*** tank, warplane**
*** match, winner**	**=> bill, program**	**<= conflict, war**	**=> town, location**

Table 1: Sample of the data set of top-40 distributionally similar word pairs produced by the *RFF*-based method of (Geffet and Dagan, 2004). Entailment judgments are marked by the arrow direction, with '*' denoting no entailment.

nyms) and the kind of tight semantic relationship that might hold between them. We formulated a lexical entailment relation that corresponds to the above mentioned substitutability criterion, and is termed *meaning entailing substitutability* (which we term here for brevity as *lexical entailment*). Given a pair of words, this relation holds if there are some contexts in which one of the words can be substituted by the other, such that the meaning of the original word can be inferred from the new one. We then proposed a new feature weighting function (*RFF*) that yields more accurate distributional similarity lists, which better approximate the lexical entailment relation. Yet, this method still applies a standard measure for distributional vector similarity (over vectors with the improved feature weights), and thus produces many loose similarities that do not correspond to entailment.

This paper explores more deeply the relationship between distributional characterization of words and lexical entailment, proposing two new hypotheses as a refinement of the distributional similarity hypothesis. The main idea is that if one word entails the other then we would expect that virtually all the characteristic context features of the entailing word will actually occur also with the entailed word.

To test this idea we developed an automatic method for testing feature inclusion between a pair of words. This algorithm combines corpus statistics with a web-based feature sampling technique. The web is utilized to overcome the data sparseness problem, so that features which are not found with one of the two words can be considered as truly distinguishing evidence.

Using the above algorithm we first tested the empirical validity of the hypotheses. Then, we demonstrated how the hypotheses can be leveraged in practice to improve the precision of automatic acquisition of the entailment relation.

2 Background

2.1 Implementations of Distributional Similarity

This subsection reviews the relevant details of earlier methods that were utilized within this paper.

In the computational setting contexts of words are represented by feature vectors. Each word *w* is represented by a feature vector, where an entry in the vector corresponds to a feature *f*. Each feature represents another word (or term) with which *w* co-occurs, and possibly specifies also the syntactic relation between the two words as in (Grefenstette, 1994; Lin, 1998; Weeds and Weir, 2003). Pado and Lapata (2003) demonstrated that using syntactic dependency-based vector space models can help distinguish among classes of different lexical relations, which seems to be more difficult for traditional "bag of words" co-occurrence-based models.

A syntactic feature is defined as a triple <*term, syntactic_relation, relation_direction*> (the direction is set to 1, if the feature is the word's modifier and to 0 otherwise). For example, given the word "company" the feature <*earnings_report, gen, 0*> (genitive) corresponds to the phrase "company's earnings report", and <*profit, pcomp, 0*> (prepositional complement) corresponds to "the profit of the company". Throughout this paper we used syntactic features generated by the Minipar dependency parser (Lin, 1993).

The value of each entry in the feature vector is determined by some weight function *weight(w,f)*, which quantifies the degree of statistical association between the feature and the corresponding word. The most widely used association weight function is (point-wise) Mutual Information (*MI*) (Church and Hanks, 1990; Lin, 1998; Dagan, 2000; Weeds et al., 2004).

Once feature vectors have been constructed, the similarity between two words is defined by some vector similarity metric. Different metrics have been used, such as weighted Jaccard (Grefenstette, 1994; Dagan, 2000), cosine (Ruge, 1992), various information theoretic measures (Lee, 1997), and the widely cited and competitive (see (Weeds and Weir, 2003)) measure of Lin (1998) for similarity between two words, *w* and *v*, defined as follows:

$$sim_{Lin}(w,v) = \frac{\sum_{f\in F(w)\cap F(v)} weight(w,f) + weight(v,f)}{\sum_{f\in F(w)} weight(w,f) + \sum_{f\in F(v)} weight(v,f)},$$

where *F(w)* and *F(v)* are the active features of the two words (positive feature weight) and the weight function is defined as *MI*. As typical for vector similarity measures, it assigns high similarity scores if many of the two word's features overlap, even though some prominent features might be disjoint. This is a major reason for getting such semantically loose similarities, like *company* - *government* and *country* - *economy*.

Investigating the output of Lin's (1998) similarity measure with respect to the above criterion in (Geffet and Dagan, 2004), we discovered that the quality of similarity scores is often hurt by inaccurate feature weights, which yield rather noisy feature vectors. Hence, we tried to improve the feature weighting function to promote those features that are most indicative of the word meaning. A new weighting scheme was defined for bootstrapping feature weights, termed *RFF* (Relative Feature Focus). First, basic similarities are generated by Lin's measure. Then, feature weights are recalculated, boosting the weights of features that characterize many of the words that are most similar to the given one[2]. As a result the most prominent features of a word are concentrated within the top-100 entries of the vector. Finally, word similarities are recalculated by Lin's metric over the vectors with the new *RFF* weights.

The lexical entailment prediction task of (Geffet and Dagan, 2004) measures how many of the top ranking similarity pairs produced by the *RFF*-based metric hold the entailment relation, in at least one direction. To this end a data set of 1,200 pairs was created, consisting of top-*N (N=40)* similar words of 30 randomly selected nouns, which were manually judged by the lexical entailment criterion. Quite high Kappa agreement values of 0.75 and 0.83 were reported, indicating that the entailment judgment task was reasonably well defined. A subset of the data set is demonstrated in Table 1.

The *RFF* weighting produced 10% precision improvement over Lin's original use of MI, suggesting the *RFF* capability to promote semantically meaningful features. However, over 47% of the word pairs in the top-40 similarities are not related by entailment, which calls for further improvement. In this paper we use the same data set [3] and the *RFF* metric as a basis for our experiments.

2.2 Predicting Semantic Inclusion

Weeds et al. (2004) attempted to refine the distributional similarity goal to predict whether one term is a generalization/specification of the other. They present a *distributional generality* concept and expect it to correlate with semantic generality. Their conjecture is that the majority of the features of the more specific word are included in the features of the more general one. They define the feature *recall* of *w* with respect to *v* as the weighted proportion of features of *v* that also appear in the vector of *w*. Then, they suggest that a hypernym would have a higher feature recall for its hyponyms (specifications), than vice versa.

However, their results in predicting the hyponymy-hyperonymy direction (71% precision) are comparable to the naïve baseline (70% precision) that simply assumes that general words are more frequent than specific ones. Possible sources of noise in their experiment could be ignoring word polysemy and data sparseness of word-feature co-occurrence in the corpus.

3 The Distributional Inclusion Hypotheses

In this paper we suggest refined versions of the distributional similarity hypothesis which relate distributional behavior with lexical entailment.

[2] In concrete terms *RFF* is defined by:

$RFF(w,f)=\sum_{v\in WS(f)\cap N(w)} sim(w,v)$,

where *sim(w,v)* is an initial approximation of the similarity space by Lin's measure, *WS(f)* is a set of words co-occurring with feature *f*, and *N(w)* is the set of the most similar words of *w* by Lin's measure.

[3] Since the original data set did not include the direction of entailment, we have enriched it by adding the judgments of entailment direction.

Extending the rationale of Weeds et al., we suggest that if the meaning of a word *v* entails another word *w* then it is expected that all the typical contexts (features) of *v* will occur also with *w*. That is, the characteristic contexts of *v* are expected to be included within all *w*'s contexts (but not necessarily amongst the most characteristic ones for *w*). Conversely, we might expect that if *v*'s characteristic contexts are included within all *w*'s contexts then it is likely that the meaning of *v* does entail *w*. Taking both directions together, lexical entailment is expected to highly correlate with characteristic feature inclusion.

Two additional observations are needed before concretely formulating these hypotheses. As explained in Section 2, word contexts should be represented by syntactic features, which are more restrictive and thus better reflect the restrained semantic meaning of the word (it is difficult to tie entailment to looser context representations, such as co-occurrence in a text window). We also notice that distributional similarity principles are intended to hold at the sense level rather than the word level, since different senses have different characteristic contexts (even though computational common practice is to work at the word level, due to the lack of robust sense annotation).

We can now define the two *distributional inclusion hypotheses*, which correspond to the two directions of inference relating distributional feature inclusion and lexical entailment. Let v_i and w_j be two word senses of the words *w* and *v*, correspondingly, and let $v_i => w_j$ denote the (directional) entailment relation between these senses. Assume further that we have a measure that determines the set of *characteristic* features for the meaning of each word sense. Then we would hypothesize:

Hypothesis I:

If $v_i => w_j$ then all the characteristic (syntactic-based) features of v_i are expected to appear with w_j.

Hypothesis II:

If all the characteristic (syntactic-based) features of v_i appear with w_j then we expect that $v_i => w_j$.

4 Word Level Testing of Feature Inclusion

To check the validity of the hypotheses we need to test feature inclusion. In this section we present an automated word-level feature inclusion testing method, termed ITA (*Inclusion Testing Algorithm*). To overcome the data sparseness problem we incorporated web-based feature sampling. Given a test pair of words, three main steps are performed, as detailed in the following subsections:

Step 1: Computing the set of characteristic features for each word.

Step 2: Testing feature inclusion for each pair, in both directions, within the given corpus data.

Step 3: Complementary testing of feature inclusion for each pair in the web.

4.1 Step 1: Corpus-based generation of characteristic features

To implement the first step of the algorithm, the *RFF* weighting function is exploited and its top-100 weighted features are taken as most characteristic for each word. As mentioned in Section 2, (Geffet and Dagan, 2004) shows that *RFF* yields high concentration of good features at the top of the vector.

4.2 Step 2: Corpus-based feature inclusion test

We first check feature inclusion in the corpus that was used to generate the characteristic feature sets. For each word pair *(w, v)* we first determine which features of *w* do co-occur with *v* in the corpus. The same is done to identify features of *v* that co-occur with *w* in the corpus.

4.3 Step 3: Complementary Web-based Inclusion Test

This step is most important to avoid inclusion misses due to the data sparseness of the corpus. A few recent works (Ravichandran and Hovy, 2002; Keller et al., 2002; Chklovski and Pantel, 2004) used the web to collect statistics on word co-occurrences. In a similar spirit, our inclusion test is completed by searching the web for the missing (non-included) features on both sides. We call this web-based technique *mutual web-sampling*. The web results are further parsed to verify matching of the feature's syntactic relationship.

We denote the subset of *w*'s features that are missing for *v* as *M(w, v)* (and equivalently *M(v, w)*). Since web sampling is time consuming we randomly sample a subset of *k* features (*k=20* in our experiments), denoted as *M(v,w,k)*.

Mutual Web-sampling Procedure:

For each pair *(w, v)* and their *k*-subsets *M(w, v, k)* and *M(v, w, k)* execute:

1. Syntactic Filtering of "Bag-of-Words" Search:

Search the web for sentences including *v* and a feature *f* from *M(w, v, k)* as "bag of words", i. e. sentences where *w* and *f* appear in any distance and in either order. Then filter out the sentences that do not match the defined syntactic relation between *f* and *v* (based on parsing). Features that co-occur with *w* in the correct syntactic relation are removed from *M(w, v, k).* Do the same search and filtering for *w* and features from *M(v, w, k).*

2. Syntactic Filtering of "Exact String" Matching:

On the missing features on both sides (which are left in *M(w, v, k)* and *M(v, w, k)* after stage 1), apply "exact string" search of the web. For this, convert the tuple *(v, f)* to a string by adding prepositions and articles where needed. For example, for *(element, <project, pcomp_of, 1>)* generate the corresponding string "element of the project" and search the web for exact matches of the string. Then validate the syntactic relationship of *f* and *v* in the extracted sentences. Remove the found features from *M(w, v, k)* and *M(v, w, k)*, respectively.

3. Missing Features Validation:

Since some of the features may be too infrequent or corpus-biased, check whether the remaining missing features do co-occur on the web with their original target words (with which they did occur in the corpus data). Otherwise, they should not be considered as valid misses and are also removed from *M(w, v, k)* and *M(v, w, k).*

Output: Inclusion in either direction holds if the corresponding set of missing features is now empty.

We also experimented with features consisting of words without syntactic relations. For example, exact string, or bag-of-words match. However, almost all the words (also non-entailing) were found with all the features of each other, even for semantically implausible combinations (e.g. a word and a feature appear next to each other but belong to different clauses of the sentence). Therefore we conclude that syntactic relation validation is very important, especially on the web, in order to avoid coincidental co-occurrences.

5 Empirical Results

To test the validity of the distributional inclusion hypotheses we performed an empirical analysis on a selected test sample using our automated testing procedure.

5.1 Data and setting

We experimented with a randomly picked test sample of about 200 noun pairs of 1,200 pairs produced by *RFF* (for details see Geffet and Dagan, 2004) under Lin's similarity scheme (Lin, 1998). The words were judged by the lexical entailment criterion (as described in Section 2). The original percentage of correct (52%) and incorrect (48%) entailments was preserved.

To estimate the degree of validity of the distributional inclusion hypotheses we decomposed each word pair of the sample *(w, v)* to two directional pairs ordered by potential entailment direction: *(w, v)* and *(v, w)*. The 400 resulting ordered pairs are used as a test set in Sections 5.2 and 5.3.

Features were computed from co-occurrences in a subset of the Reuters corpus of about 18 million words. For the web feature sampling the maximal number of web samples for each query (word - feature) was set to 3,000 sentences.

5.2 Automatic Testing the Validity of the Hypotheses at the Word Level

The test set of 400 ordered pairs was examined in terms of entailment (according to the manual judgment) and feature inclusion (according to the ITA algorithm), as shown in Table 2.

According to Hypothesis I we expect that a pair *(w, v)* that satisfies entailment will also preserve feature inclusion. On the other hand, by Hypothesis II if all the features of *w* are included by *v* then we expect that *w* entails *v*.

Entailment \ Inclusion	+	-
+	**97**	**16**
-	**42**	**245**

Table 2: Distribution of 400 entailing/non-entailing ordered pairs that hold/do not hold feature inclusion at the *word* level.

spread – gap (mutually entail each other) ***<weapon, pcomp_of>*** The Committee was discussing the Programme of the "Big Eight," aimed against **spread of weapon** of mass destruction.
town – area ("town" entails "area") ***<cooperation, pcomp_for>*** This is a promising **area for cooperation** and exchange of experiences.
capital – town ("capital" entails "town") ***<flow, nn>*** Offshore financial centers affect cross-border **capital flow** in China.

Table 3: Examples of ambiguity of entailment-related words, where the disjoint features belong to a different sense of the word.

We observed that Hypothesis I is better attested by our data than the second hypothesis. Thus 86% (97 out of 113) of the entailing pairs fulfilled the inclusion condition. Hypothesis II holds for approximately 70% (97 of 139) of the pairs for which feature inclusion holds. In the next section we analyze the cases of violation of both hypotheses and find that the first hypothesis held to an almost perfect extent with respect to word senses.

It is also interesting to note that thanks to the web-sampling procedure over 90% of the non-included features in the corpus were found on the web, while most of the missing features (in the web) are indeed semantically implausible.

5.3 Manual Sense Level Testing of Hypotheses Validity

Since our data was not sense tagged, the automatic validation procedure could only test the hypotheses at the word level. In this section our goal is to analyze the findings of our empirical test at the word sense level as our hypotheses were defined for senses. Basically, two cases of hypotheses invalidity were detected:

Case 1: Entailments with non-included features (violation of Hypothesis I);

Case 2: Feature Inclusion for non-entailments (violation of Hypothesis II).

Entailment \ Inclusion	+	-
+	**111**	**2**
-	**42**	**245**

Table 4: Distribution of the entailing/non-entailing ordered pairs that hold/do not hold feature inclusion at the *sense* level.

At the word level we observed 14% invalid pairs of the first case and 30% of the second case. However, our manual analysis shows, that over 90% of the first case pairs were due to a different sense of one of the entailing word, e.g. *capital - town* (*capital* as money) and *spread - gap* (*spread* as distribution) (Table 3). Note that ambiguity of the entailed word does not cause errors (like *town – area*, *area* as domain) (Table 3). Thus the first hypothesis holds at the sense level for over 98% of the cases (Table 4).

Two remaining invalid instances of the first case were due to the web sampling method limitations and syntactic parsing filtering mistakes, especially for some less characteristic and infrequent features captured by *RFF*. Thus, in virtually all the examples tested in our experiment Hypothesis I was valid.

We also explored the second case of invalid pairs: non-entailing words that pass the feature inclusion test. After sense based analysis their percentage was reduced slightly to 27.4%. Three possible reasons were discovered. First, there are words with features typical to the general meaning of the domain, which tend to be included by many other words of this domain, like *valley – town*. The features of *valley* ("eastern valley", "central valley", "attack in valley", "industry of the valley") are not discriminative enough to be distinguished from *town*, as they are all characteristic to any geographic location.

Method	*Precision*	*Recall*	*F1*
ITA-20	0.700	0.875	0.777
ITA-40	0.740	0.846	0.789
RFF-top-40	0.520	1.000	0.684
RFF-top-26	0.561	0.701	0.624

Table 5: Comparative results of using the filter, with 20 and 40 feature sampling, compared to *RFF* top-40 and *RFF* top-26 similarities. ITA-20 and ITA-40 denote the web-sampling method with 20 and random 40 features, respectively.

The second group consists of words that can be entailing, but only in a context-dependent (anaphoric) manner rather than ontologically. For example, *government* and *neighbour*, while *neighbour* is used in the meaning of *"neighbouring (country) government"*. Finally, sometimes one or both of the words are abstract and general enough and also highly ambiguous to appear with a wide range of features on the web, like *element (violence – element*, with all the tested features of *violence* included by *element).*

To prevent occurrences of the second case more characteristic and discriminative features should be provided. For this purpose features extracted from the web, which are not domain-biased (like features from the corpus) and multi-word features may be helpful. Overall, though, there might be inherent cases that invalidate Hypothesis II.

6 Improving Lexical Entailment Prediction by ITA (Inclusion Testing Algorithm)

In this section we show that ITA can be practically used to improve the (non-directional) lexical entailment prediction task described in Section 2. Given the output of the distributional similarity method, we employ ITA at the word level to filter out non-entailing pairs. Word pairs that satisfy feature inclusion of all *k* features (at least in one direction) are claimed as entailing.

The same test sample of 200 word pairs mentioned in Section 5.1 was used in this experiment. The results were compared to *RFF* under Lin's similarity scheme (*RFF*-top-40 in Table 5).

Precision was significantly improved, filtering out 60% of the incorrect pairs. On the other hand, the relative recall (considering *RFF* recall as 100%) was only reduced by 13%, consequently leading to a better relative F1, when considering the *RFF*-top-40 output as 100% recall (Table 5).

Since our method removes about 35% of the original top-40 *RFF* output, it was interesting to compare our results to simply cutting off the 35% of the lowest ranked *RFF* words (top-26). The comparison to the baseline (*RFF*-top-26 in Table 5) showed that ITA filters the output much better than just cutting off the lowest ranking similarities.

We also tried a couple of variations on feature sampling for the web-based procedure. In one of our preliminary experiments we used the top-*k* *RFF* features instead of random selection. But we observed that top ranked *RFF* features are less discriminative than the random ones due to the nature of the *RFF* weighting strategy, which promotes features *shared* by many similar words. Then, we attempted doubling the sampling to 40 random features. As expected the recall was slightly decreased, while precision was increased by over 5%. In summary, the behavior of ITA sampling of *k=20* and *k=40* features is closely comparable (ITA-20 and ITA-40 in Table 5, respectively)[4].

7 Conclusions and Future Work

The main contributions of this paper were:

1. We defined two Distributional Inclusion Hypotheses that associate feature inclusion with lexical entailment at the word sense level. The Hypotheses were proposed as a refinement for Harris' Distributional hypothesis and as an extension to the classic distributional similarity scheme.
2. To estimate the empirical validity of the defined hypotheses we developed an automatic *inclusion testing algorithm* (ITA). The core of the algorithm is a web-based feature inclusion testing procedure, which helped significantly to compensate for data sparseness.
3. Then a thorough analysis of the data behavior with respect to the proposed hypotheses was conducted. The first hypothesis was almost fully attested by the data, particularly at the sense level, while the second hypothesis did not fully hold.
4. Motivated by the empirical analysis we proposed to employ ITA for the practical task of improving lexical entailment acquisition. The algorithm was applied as a filtering technique on the distributional similarity (*RFF*) output. We ob-

[4] The ITA-40 sampling fits the analysis from section 5.2 and 5.3 as well.

tained 17% increase of precision and succeeded to improve relative F1 by 15% over the baseline.

Although the results were encouraging our manual data analysis shows that we still have to handle word ambiguity. In particular, this is important in order to be able to learn the direction of entailment.

To achieve better precision we need to increase feature discriminativeness. To this end syntactic features may be extended to contain more than one word, and ways for automatic extraction of features from the web (rather than from a corpus) may be developed. Finally, further investigation of combining the distributional and the co-occurrence pattern-based approaches over the web is desired.

Acknowledgement

We are grateful to Shachar Mirkin for his help in implementing the web-based sampling procedure heavily employed in our experiments. We thank Idan Szpektor for providing the infrastructure system for web-based data extraction.

References

Chklovski, Timothy and Patrick Pantel. 2004. VERBOCEAN: Mining the Web for Fine-Grained Semantic Verb Relations. In Proc. of EMNLP-04. Barcelona, Spain.

Church, Kenneth W. and Hanks Patrick. 1990. Word association norms, mutual information, and Lexicography. Computational Linguistics, 16(1), pp. 22–29.

Dagan, Ido. 2000. Contextual Word Similarity, in Rob Dale, Hermann Moisl and Harold Somers (Eds.), Handbook of Natural Language Processing, Marcel Dekker Inc, 2000, Chapter 19, pp. 459-476.

Dagan, Ido, Oren Glickman and Bernardo Magnini. 2005. The PASCAL Recognizing Textual Entailment Challenge. In Proc. of the PASCAL Challenges Workshop for Recognizing Textual Entailment. Southampton, U.K.

Geffet, Maayan and Ido Dagan, 2004. Feature Vector Quality and Distributional Similarity. In Proc. of Coling-04. Geneva. Switzerland.

Grefenstette, Gregory. 1994. Exploration in Automatic Thesaurus Discovery. Kluwer Academic Publishers.

Harris, Zelig S. Mathematical structures of language. Wiley, 1968.

Hearst, Marti. 1992. Automatic acquisition of hyponyms from large text corpora. In Proc. of COLING-92. Nantes, France.

Keller, Frank, Maria Lapata, and Olga Ourioupina. 2002. Using the Web to Overcome Data Sparseness. In Jan Hajic and Yuji Matsumoto, eds., In Proc. of EMNLP-02. Philadelphia, PA.

Lee, Lillian. 1997. Similarity-Based Approaches to Natural Language Processing. Ph.D. thesis, Harvard University, Cambridge, MA.

Lin, Dekang. 1993. Principle-Based Parsing without Overgeneration. In Proc. of ACL-93. Columbus, Ohio.

.

Lin, Dekang. 1998. Automatic Retrieval and Clustering of Similar Words. In Proc. of COLING–ACL98, Montreal, Canada.

Moldovan, Dan, Badulescu, A., Tatu, M., Antohe, D., and Girju, R. 2004. Models for the semantic classification of noun phrases. In Proc. of HLT/NAACL-2004 Workshop on Computational Lexical Semantics. Boston.

Pado, Sebastian and Mirella Lapata. 2003. Constructing semantic space models from parsed corpora. In Proc. of ACL-03, Sapporo, Japan.

Pantel, Patrick and Dekang Lin. 2002. Discovering Word Senses from Text. In Proc. of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-02). Edmonton, Canada.

Pereira, Fernando, Tishby Naftali, and Lee Lillian. 1993. Distributional clustering of English words. In Proc. of ACL-93. Columbus, Ohio.

Ravichandran, Deepak and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In Proc. of ACL-02. Philadelphia, PA.

Ruge, Gerda. 1992. Experiments on linguistically-based term associations. Information Processing & Management, 28(3), pp. 317–332.

Weeds, Julie and David Weir. 2003. A General Framework for Distributional Similarity. In Proc. of EMNLP-03. Sapporo, Japan.

Weeds, Julie, D. Weir, D. McCarthy. 2004. Characterizing Measures of Lexical Distributional Similarity. In Proc. of Coling-04. Geneva, Switzerland.

Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales

Bo Pang[1,3] and **Lillian Lee**[1,2,3]
(1) Department of Computer Science, Cornell University
(2) Language Technologies Institute, Carnegie Mellon University
(3) Computer Science Department, Carnegie Mellon University

Abstract

We address the *rating-inference problem*, wherein rather than simply decide whether a review is “thumbs up” or “thumbs down”, as in previous sentiment analysis work, one must determine an author’s evaluation with respect to a multi-point scale (e.g., one to five “stars”). This task represents an interesting twist on standard multi-class text categorization because there are several different degrees of similarity between class labels; for example, “three stars” is intuitively closer to “four stars” than to “one star”.

We first evaluate human performance at the task. Then, we apply a meta-algorithm, based on a *metric labeling* formulation of the problem, that alters a given n-ary classifier’s output in an explicit attempt to ensure that similar items receive similar labels. We show that the meta-algorithm can provide significant improvements over both multi-class and regression versions of SVMs when we employ a novel similarity measure appropriate to the problem.

1 Introduction

There has recently been a dramatic surge of interest in *sentiment analysis*, as more and more people become aware of the scientific challenges posed and the scope of new applications enabled by the processing of subjective language. (The papers collected by Qu, Shanahan, and Wiebe (2004) form a representative sample of research in the area.) Most prior work on the specific problem of categorizing expressly opinionated text has focused on the binary distinction of positive vs. negative (Turney, 2002; Pang, Lee, and Vaithyanathan, 2002; Dave, Lawrence, and Pennock, 2003; Yu and Hatzivassiloglou, 2003). But it is often helpful to have more information than this binary distinction provides, especially if one is ranking items by recommendation or comparing several reviewers’ opinions: example applications include collaborative filtering and deciding which conference submissions to accept.

Therefore, in this paper we consider generalizing to finer-grained *scales*: rather than just determine whether a review is “thumbs up” or not, we attempt to infer the author’s implied numerical rating, such as “three stars” or “four stars”. Note that this differs from identifying opinion *strength* (Wilson, Wiebe, and Hwa, 2004): rants and raves have the same strength but represent opposite evaluations, and referee forms often allow one to indicate that one is very confident (high strength) that a conference submission is mediocre (middling rating). Also, our task differs from *ranking* not only because one can be given a single item to classify (as opposed to a set of items to be ordered relative to one another), but because there are settings in which classification is harder than ranking, and vice versa.

One can apply standard n-ary classifiers or regression to this *rating-inference problem*; independent work by Koppel and Schler (2005) considers such methods. But an alternative approach that explicitly incorporates information about item similarities together with label similarity information (for instance, "one star" is closer to "two stars" than to "four stars") is to think of the task as one of *metric labeling* (Kleinberg and Tardos, 2002), where label relations are encoded via a distance metric. This observation yields a meta-algorithm, applicable to both semi-supervised (via graph-theoretic techniques) and supervised settings, that alters a given n-ary classifier's output so that similar items tend to be assigned similar labels.

In what follows, we first demonstrate that humans can discern relatively small differences in (hidden) evaluation scores, indicating that rating inference is indeed a meaningful task. We then present three types of algorithms — one-vs-all, regression, and metric labeling — that can be distinguished by how explicitly they attempt to leverage similarity between items and between labels. Next, we consider what item similarity measure to apply, proposing one based on the *positive-sentence percentage*. Incorporating this new measure within the metric-labeling framework is shown to often provide significant improvements over the other algorithms.

We hope that some of the insights derived here might apply to other scales for text classifcation that have been considered, such as clause-level opinion strength (Wilson, Wiebe, and Hwa, 2004); affect types like disgust (Subasic and Huettner, 2001; Liu, Lieberman, and Selker, 2003); reading level (Collins-Thompson and Callan, 2004); and urgency or criticality (Horvitz, Jacobs, and Hovel, 1999).

2 Problem validation and formulation

We first ran a small pilot study on human subjects in order to establish a rough idea of what a reasonable classification granularity is: if even people cannot accurately infer labels with respect to a five-star scheme with half stars, say, then we cannot expect a learning algorithm to do so. Indeed, some potential obstacles to accurate rating inference include lack of calibration (e.g., what an understated author intends as high praise may seem lukewarm), author inconsistency at assigning fine-grained ratings, and ratings not entirely supported by the text[1].

Rating diff.	Pooled	Subject 1	Subject 2
3 or more	100%	100% (35)	100% (15)
2 (e.g., 1 star)	83%	77% (30)	100% (11)
1 (e.g., $\frac{1}{2}$ star)	69%	65% (57)	90% (10)
0	55%	47% (15)	80% (5)

Table 1: Human accuracy at determining relative positivity. Rating differences are given in "notches". Parentheses enclose the number of pairs attempted.

For data, we first collected Internet movie reviews in English from four authors, removing explicit rating indicators from each document's text automatically. Now, while the obvious experiment would be to ask subjects to guess the rating that a review represents, doing so would force us to specify a fixed rating-scale granularity in advance. Instead, we examined people's ability to discern *relative differences*, because by varying the rating differences represented by the test instances, we can evaluate multiple granularities in a single experiment. Specifically, at intervals over a number of weeks, we authors (a non-native and a native speaker of English) examined pairs of reviews, attemping to determine whether the first review in each pair was (1) more positive than, (2) less positive than, or (3) as positive as the second. The texts in any particular review pair were taken from the same author to factor out the effects of cross-author divergence.

As Table 1 shows, both subjects performed perfectly when the rating separation was at least 3 "notches" in the original scale (we define a notch as a half star in a four- or five-star scheme and 10 points in a 100-point scheme). Interestingly, although human performance drops as rating difference decreases, even at a one-notch separation, both subjects handily outperformed the random-choice baseline of 33%. However, there was large variation in accuracy between subjects.[2]

[1]For example, the critic Dennis Schwartz writes that "sometimes the review itself [indicates] the letter grade should have been higher or lower, as the review might fail to take into consideration my overall impression of the film — which I hope to capture in the grade" (http://www.sover.net/~ozus/cinema.htm).

[2]One contributing factor may be that the subjects viewed disjoint document sets, since we wanted to maximize experimental coverage of the types of document pairs within each difference class. We thus cannot report inter-annotator agreement,

Because of this variation, we defined two different classification regimes. From the evidence above, a **three-class** task (categories 0, 1, and 2 — essentially "negative", "middling", and "positive", respectively) seems like one that most people would do quite well at (but we should not assume 100% human accuracy: according to our one-notch results, people may misclassify borderline cases like 2.5 stars). Our study also suggests that people could do at least fairly well at distinguishing full stars in a zero- to four-star scheme. However, when we began to construct five-category datasets for each of our four authors (see below), we found that in each case, either the most negative or the most positive class (but not both) contained only about 5% of the documents. To make the classes more balanced, we folded these minority classes into the adjacent class, thus arriving at a **four-class** problem (categories 0-3, increasing in positivity). Note that the four-class problem seems to offer more possibilities for leveraging class relationship information than the three-class setting, since it involves more class pairs. Also, even the two-category version of the rating-inference problem for movie reviews has proven quite challenging for many automated classification techniques (Pang, Lee, and Vaithyanathan, 2002; Turney, 2002).

We applied the above two labeling schemes to a **scale dataset**[3] containing four corpora of movie reviews. All reviews were automatically preprocessed to remove both explicit rating indicators and objective sentences; the motivation for the latter step is that it has previously aided positive vs. negative classification (Pang and Lee, 2004). All of the 1770, 902, 1307, or 1027 documents in a given corpus were written by the same author. This decision facilitates interpretation of the results, since it factors out the effects of different choices of methods for calibrating authors' scales.[4] We point out that it is possible to gather author-specific information in some practical applications: for instance, systems that use selected authors (e.g., the Rotten Tomatoes movie-review website — where, we note, not all authors provide explicit ratings) could require that someone submit rating-labeled samples of newly-admitted authors' work. Moreover, our results at least partially generalize to mixed-author situations (see Section 5.2).

3 Algorithms

Recall that the problem we are considering is multi-category classification in which the labels can be naturally mapped to a metric space (e.g., points on a line); for simplicity, we assume the distance metric $d(\ell, \ell') = |\ell - \ell'|$ throughout. In this section, we present three approaches to this problem in order of increasingly explicit use of pairwise similarity information between items and between labels. In order to make comparisons between these methods meaningful, we base all three of them on Support Vector Machines (SVMs) as implemented in Joachims' (1999) SVM^{light} package.

3.1 One-vs-all

The standard SVM formulation applies only to binary classification. *One-vs-all* (OVA) (Rifkin and Klautau, 2004) is a common extension to the n-ary case. Training consists of building, for each label ℓ, an SVM binary classifier distinguishing label ℓ from "not-ℓ". We consider the final output to be a label preference function $\pi^{\text{ova}}(x, \ell)$, defined as the signed distance of (test) item x to the ℓ side of the ℓ vs. not-ℓ decision plane.

Clearly, OVA makes no explicit use of pairwise label or item relationships. However, it can perform well if each class exhibits sufficiently distinct language; see Section 4 for more discussion.

3.2 Regression

Alternatively, we can take a *regression* perspective by assuming that the labels come from a discretization of a continuous function g mapping from the

but since our goal is to recover a reviewer's "true" recommendation, reader-author agreement is more relevant.

While another factor might be degree of English fluency, in an informal experiment (six subjects viewing the same three pairs), native English speakers made the only two errors.

[3]Available at http://www.cs.cornell.edu/People/pabo/movie-review-data as scale dataset v1.0.

[4]From the Rotten Tomatoes website's FAQ: "star systems are not consistent between critics. For critics like Roger Ebert and James Berardinelli, 2.5 stars or lower out of 4 stars is always negative. For other critics, 2.5 stars can either be positive or negative. Even though Eric Lurio uses a 5 star system, his grading is very relaxed. So, 2 stars can be positive." Thus, calibration may sometimes require strong familiarity with the authors involved, as anyone who has ever needed to reconcile conflicting referee reports probably knows.

feature space to a metric space.[5] If we choose g from a family of sufficiently "gradual" functions, then similar items necessarily receive similar labels. In particular, we consider *linear, ε-insensitive* SVM regression (Vapnik, 1995; Smola and Schölkopf, 1998); the idea is to find the hyperplane that best fits the training data, but where training points whose labels are within distance ε of the hyperplane incur no loss. Then, for (test) instance x, the label preference function $\pi^{\text{reg}}(x, \ell)$ is the negative of the distance between ℓ and the value predicted for x by the fitted hyperplane function.

Wilson, Wiebe, and Hwa (2004) used SVM regression to classify clause-level strength of opinion, reporting that it provided lower accuracy than other methods. However, independently of our work, Koppel and Schler (2005) found that applying linear regression to classify documents (in a different corpus than ours) with respect to a three-point rating scale provided greater accuracy than OVA SVMs and other algorithms.

3.3 Metric labeling

Regression *implicitly* encodes the "similar items, similar labels" heuristic, in that one can restrict consideration to "gradual" functions. But we can also think of our task as a *metric labeling* problem (Kleinberg and Tardos, 2002), a special case of the maximum *a posteriori* estimation problem for Markov random fields, to *explicitly* encode our desideratum. Suppose we have an initial label preference function $\pi(x, \ell)$, perhaps computed via one of the two methods described above. Also, let d be a distance metric on labels, and let $nn_k(x)$ denote the k nearest neighbors of item x according to some item-similarity function sim. Then, it is quite natural to pose our problem as finding a mapping of instances x to labels ℓ_x (respecting the original labels of the training instances) that minimizes

$$\sum_{x \in \text{test}} \left[-\pi(x, \ell_x) + \alpha \sum_{y \in nn_k(x)} f(d(\ell_x, \ell_y))\, sim(x, y) \right],$$

where f is monotonically increasing (we chose $f(d) = d$ unless otherwise specified) and α is a trade-off and/or scaling parameter. (The inner summation is familiar from work in *locally-weighted learning*[6] (Atkeson, Moore, and Schaal, 1997).) In a sense, we are using explicit item and label similarity information to increasingly penalize the initial classifier as it assigns more divergent labels to similar items.

In this paper, we only report supervised-learning experiments in which the nearest neighbors for any given test item were drawn from the training set alone. In such a setting, the labeling decisions for different test items are independent, so that solving the requisite optimization problem is simple.

Aside: transduction The above formulation also allows for *transductive* semi-supervised learning as well, in that we could allow nearest neighbors to come from both the training and test sets. We intend to address this case in future work, since there are important settings in which one has a small number of labeled reviews and a large number of unlabeled reviews, in which case considering similarities between unlabeled texts could prove quite helpful. In full generality, the corresponding multi-label optimization problem is intractable, but for many families of f functions (e.g., convex) there exist practical exact or approximation algorithms based on techniques for finding *minimum s-t cuts* in graphs (Ishikawa and Geiger, 1998; Boykov, Veksler, and Zabih, 1999; Ishikawa, 2003). Interestingly, previous sentiment analysis research found that a minimum-cut formulation for the binary subjective/objective distinction yielded good results (Pang and Lee, 2004). Of course, there are many other related semi-supervised learning algorithms that we would like to try as well; see Zhu (2005) for a survey.

4 Class struggle: finding a label-correlated item-similarity function

We need to specify an item similarity function sim to use the metric-labeling formulation described in Section 3.3. We could, as is commonly done, employ a term-overlap-based measure such as the cosine between term-frequency-based document vectors (henceforth "TO(cos)"). However, Table 2

[5]We discuss the *ordinal* regression variant in Section 6.

[6]If we ignore the $\pi(x, \ell)$ term, different choices of f correspond to different versions of nearest-neighbor learning, e.g., majority-vote, weighted average of labels, or weighted median of labels.

	Label difference: 1	2	3
Three-class data	37%	33%	—
Four-class data	34%	31%	30%

Table 2: Average over authors and class pairs of between-class vocabulary overlap as the class labels of the pair grow farther apart.

shows that in aggregate, the vocabularies of distant classes overlap to a degree surprisingly similar to that of the vocabularies of nearby classes. Thus, item similarity as measured by TO(cos) may not correlate well with similarity of the item's true labels.

We can potentially develop a more useful similarity metric by asking ourselves what, intuitively, accounts for the label relationships that we seek to exploit. A simple hypothesis is that ratings can be determined by the *positive-sentence percentage (PSP)* of a text, i.e., the number of positive sentences divided by the number of subjective sentences. (Term-based versions of this premise have motivated much sentiment-analysis work for over a decade (Das and Chen, 2001; Tong, 2001; Turney, 2002).) But counterexamples are easy to construct: reviews can contain off-topic opinions, or recount many positive aspects before describing a fatal flaw.

We therefore tested the hypothesis as follows. To avoid the need to hand-label sentences as positive or negative, we first created a *sentence polarity dataset*[7] consisting of 10,662 movie-review "snippets" (a striking extract usually one sentence long) downloaded from www.rottentomatoes.com; each snippet was labeled with its source review's label (positive or negative) as provided by Rotten Tomatoes. Then, we trained a Naive Bayes classifier on this data set and applied it to our scale dataset to identify the positive sentences (recall that objective sentences were already removed).

Figure 1 shows that all four authors tend to exhibit a higher PSP when they write a more positive review, and we expect that most typical reviewers would follow suit. Hence, PSP appears to be a promising basis for computing document similarity for our rating-inference task. In particular, we defined $\overrightarrow{\mathrm{PSP}(x)}$ to be the two-dimensional vector $(\mathrm{PSP}(x), 1 - \mathrm{PSP}(x))$, and then set the item-similarity function required by the metric-labeling optimization function (Section 3.3) to $sim(x, y) = \cos\left(\overrightarrow{\mathrm{PSP}(x)}, \overrightarrow{\mathrm{PSP}(y)}\right)$.[8]

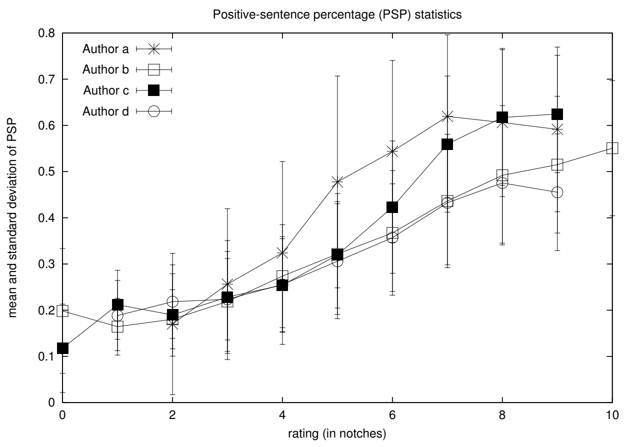


Figure 1: Average and standard deviation of PSP for reviews expressing different ratings.

But before proceeding, we note that it is possible that similarity information might yield no extra benefit at all. For instance, we don't need it if we can reliably identify each class just from some set of distinguishing terms. If we define such terms as frequent ones ($n \geq 20$) that appear in a single class 50% or more of the time, then we do find many instances; some examples for one author are: "meaningless", "disgusting" (class 0); "pleasant", "uneven" (class 1); and "oscar", "gem" (class 2) for the three-class case, and, in the four-class case, "flat", "tedious" (class 1) versus "straightforward", "likeable" (class 2). Some unexpected distinguishing terms for this author are "lion" for class 2 (three-class case), and for class 2 in the four-class case, "jennifer", for a wide variety of Jennifers.

5 Evaluation

This section compares the accuracies of the approaches outlined in Section 3 on the four corpora comprising our scale dataset. (Results using L_1 error were qualitatively similar.) Throughout, when

[7] Available at http://www.cs.cornell.edu/People/pabo/movie-review-data as sentence polarity dataset v1.0.

[8] While admittedly we initially chose this function because it was convenient to work with cosines, *post hoc* analysis revealed that the corresponding metric space "stretched" certain distances in a useful way.

we refer to something as "significant", we mean statistically so with respect to the paired t-test, $p < .05$.

The results that follow are based on SVM^{light}'s default parameter settings for SVM regression and OVA. Preliminary analysis of the effect of varying the regression parameter ε in the four-class case revealed that the default value was often optimal.

The notation "A+B" denotes metric labeling where method A provides the initial label preference function π and B serves as similarity measure. To train, we first select the meta-parameters k and α by running 9-fold cross-validation within the training set. Fixing k and α to those values yielding the best performance, we then re-train A (but with SVM parameters fixed, as described above) on the whole training set. At test time, the nearest neighbors of each item are also taken from the full training set.

5.1 Main comparison

Figure 2 summarizes our average 10-fold cross-validation accuracy results. We first observe from the plots that all the algorithms described in Section 3 always definitively outperform the simple baseline of predicting the majority class, although the improvements are smaller in the four-class case. Incidentally, the data was distributed in such a way that the absolute performance of the baseline itself does not change much between the three- and four-class case (which implies that the three-class datasets were relatively more balanced); and Author c's datasets seem noticeably easier than the others.

We now examine the effect of implicitly using label and item similarity. In the four-class case, regression performed better than OVA (significantly so for two authors, as shown in the righthand table); but for the three-category task, OVA significantly outperforms regression for all four authors. One might initially interprete this "flip" as showing that in the four-class scenario, item and label similarities provide a richer source of information relative to class-specific characteristics, especially since for the non-majority classes there is less data available; whereas in the three-class setting the categories are better modeled as quite distinct entities.

However, the three-class results for metric labeling on top of OVA and regression (shown in Figure 2 by black versions of the corresponding icons) show that employing explicit similarities always improves results, often to a significant degree, and yields the best overall accuracies. Thus, we *can* in fact effectively exploit similarities in the three-class case. Additionally, in both the three- and four- class scenarios, metric labeling often brings the performance of the weaker base method up to that of the stronger one (as indicated by the "disappearance" of upward triangles in corresponding table rows), and never hurts performance significantly.

In the four-class case, metric labeling and regression seem roughly equivalent. One possible interpretation is that the relevant structure of the problem is already captured by linear regression (and perhaps a different kernel for regression would have improved its three-class performance). However, according to additional experiments we ran in the four-class situation, the test-set-optimal parameter settings for metric labeling would have produced significant improvements, indicating there may be greater potential for our framework. At any rate, we view the fact that metric labeling performed quite well for both rating scales as a definitely positive result.

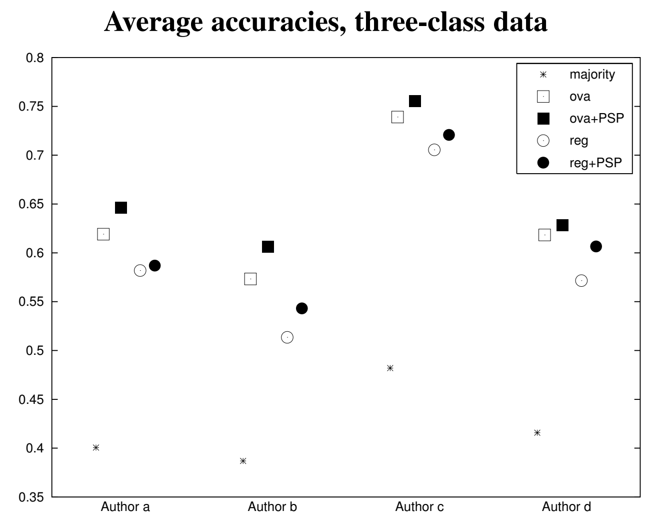
5.2 Further discussion

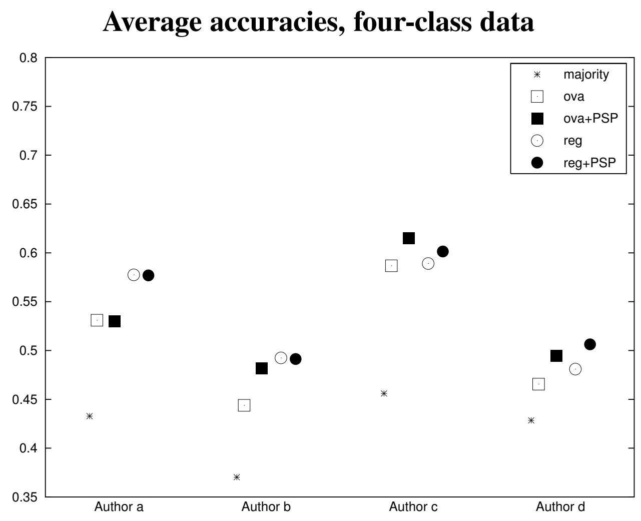
Q: Metric labeling looks like it's just combining SVMs with nearest neighbors, and classifier combination often improves performance. Couldn't we get the same kind of results by combining SVMs with any other reasonable method?

A: No. For example, if we take the strongest base SVM method for initial label preferences, but replace PSP with the term-overlap-based cosine (TO(cos)), performance often drops significantly. This result, which is in accordance with Section 4's data, suggests that choosing an item similarity function that correlates well with label similarity is important. (ova+PSP ◁◁◁◁ ova+TO(cos) [3c]; reg+PSP ◁ reg+TO(cos) [4c])

Q: Could you explain that notation, please?

A: Triangles point toward the significantly better algorithm for some dataset. For instance, "M ◁◁▷ N [3c]" means, "In the 3-class task, method M is significantly better than N for two author datasets and significantly worse for one dataset (so the algorithms were statistically indistinguishable on the remaining dataset)". When the algorithms being compared are statistically indistinguishable on





Average ten-fold cross-validation accuracies. Open icons: SVMs in either one-versus-all (square) or regression (circle) mode; dark versions: metric labeling using the corresponding SVM together with the positive-sentence percentage (PSP). The y-axes of the two plots are aligned.

Significant differences, three-class data

	ova	ova+PSP	reg	reg+PSP
	a b c d	a b c d	a b c d	a b c d
ova		△ △ △ ·	◁ ◁ ◁ ◁	· ◁ · ·
ova+PSP	◀ ◀ ◀ ·		◁ ◁ ◁ ◁	◁ ◁ ◁ ·
reg	△ △ △ △	△ △ △ △		· △ · △
reg+PSP	· △ · ·	△ △ △ ·	· ◀ · ◀	

Significant differences, four-class data

	ova	ova+PSP	reg	reg+PSP
	a b c d	a b c d	a b c d	a b c d
ova		· △ △ △	△ △ · ·	△ · · △
ova+PSP	· ◀ ◀ ◀		△ · · ·	△ · · ·
reg	◁ ◁ · ·	◁ · · ·		· · · ·
reg+PSP	◁ · · ◁	◁ · · ·	• • • •	

Triangles point towards significantly better algorithms for the results plotted above. Specifically, if the difference between a row and a column algorithm for a given author dataset (a, b, c, or d) is significant, a triangle points to the better one; otherwise, a dot (.) is shown. Dark icons highlight the effect of adding PSP information via metric labeling.

Figure 2: Results for main experimental comparisons.

all four datasets (the "no triangles" case), we indicate this with an equals sign ("=").

Q: Thanks. Doesn't Figure 1 show that the positive-sentence percentage would be a good classifier even in isolation, so metric labeling isn't necessary?
A: No. Predicting class labels directly from the PSP value via trained thresholds isn't as effective (ova+PSP ◁◁◁◁ threshold PSP [3c]; reg+PSP ◁◁ threshold PSP [4c]).

Alternatively, we could use only the PSP component of metric labeling by setting the label preference function to the constant function 0, but even with *test-set-optimal* parameter settings, doing so underperforms the *trained* metric labeling algorithm with access to an initial SVM classifier (ova+PSP ◁◁◁◁ 0+PSP* [3c]; reg+PSP ◁◁ 0+PSP* [4c]).

Q: What about using PSP as one of the features for input to a standard classifier?
A: Our focus is on investigating the utility of similarity information. In our particular rating-inference setting, it so happens that the basis for our pairwise similarity measure can be incorporated as an

item-specific feature, but we view this as a tangential issue. That being said, preliminary experiments show that metric labeling can be better, barely (for test-set-optimal parameter settings for both algorithms: significantly better results for one author, four-class case; statistically indistinguishable otherwise), although one needs to determine an appropriate weight for the PSP feature to get good performance.

Q: You defined the "metric transformation" function f as the identity function $f(d) = d$, imposing greater loss as the distance between labels assigned to two similar items increases. Can you do just as well if you penalize all non-equal label assignments by the same amount, or does the distance between labels really matter?
A: You're asking for a comparison to the *Potts model*, which sets f to the function $\hat{f}(d) = 1$ if $d > 0$, 0 otherwise. In the one setting in which there is a significant difference between the two, the Potts model does worse (ova+PSP $\triangleleft$ ova$\hat{+}$PSP [3c]). Also, employing the Potts model generally leads to fewer significant improvements over a chosen base method (compare Figure 2's tables with: reg$\hat{+}$PSP $\triangleleft$ reg [3c]; ova$\hat{+}$PSP $\triangleleft\triangleleft$ ova [3c]; ova$\hat{+}$PSP = ova [4c]; but note that reg$\hat{+}$PSP $\triangleleft$ reg [4c]). We note that optimizing the Potts model in the multi-label case is NP-hard, whereas the optimal metric labeling with the identity metric-transformation function can be efficiently obtained (see Section 3.3).

Q: Your datasets had many labeled reviews and only one author each. Is your work relevant to settings with many authors but very little data for each?
A: As discussed in Section 2, it can be quite difficult to properly calibrate different authors' scales, since the same number of "stars" even within what is ostensibly the same rating system can mean different things for different authors. But since you ask: we temporarily turned a blind eye to this serious issue, creating a collection of 5394 reviews by 496 authors with at most 80 reviews per author, where we pretended that our rating conversions mapped correctly into a universal rating scheme. Preliminary results on this dataset were actually comparable to the results reported above, although since we are not confident in the class labels themselves, more work is needed to derive a clear analysis of this setting. (Abusing notation, since we're already playing fast and loose: [3c]: baseline 52.4%, reg 61.4%, reg+PSP 61.5%, ova (65.4%) $\triangleright$ ova+PSP (66.3%); [4c]: baseline 38.8%, reg (51.9%) $\triangleright$ reg+PSP (52.7%), ova (53.8%) $\triangleright$ ova+PSP (54.6%))

In future work, it would be interesting to determine author-independent characteristics that can be used on (or suitably adapted to) data for specific authors.

Q: How about trying —
A: —Yes, there are many alternatives. A few that we tested are described in the Appendix, and we propose some others in the next section. We should mention that we have not yet experimented with *all-vs.-all* (AVA), another standard binary-to-multi-category classifier conversion method, because we wished to focus on the effect of omitting pairwise information. In independent work on 3-category rating inference for a different corpus, Koppel and Schler (2005) found that regression outperformed AVA, and Rifkin and Klautau (2004) argue that in principle OVA should do just as well as AVA. But we plan to try it out.

6 Related work and future directions

In this paper, we addressed the rating-inference problem, showing the utility of employing label similarity and (appropriate choice of) item similarity — either implicitly, through regression, or explicitly and often more effectively, through metric labeling.

In the future, we would like to apply our methods to other scale-based classification problems, and explore alternative methods. Clearly, varying the kernel in SVM regression might yield better results. Another choice is *ordinal regression* (McCullagh, 1980; Herbrich, Graepel, and Obermayer, 2000), which only considers the ordering on labels, rather than any explicit distances between them; this approach could work well if a good metric on labels is lacking. Also, one could use mixture models (e.g., combine "positive" and "negative" language models) to capture class relationships (McCallum, 1999; Schapire and Singer, 2000; Takamura, Matsumoto, and Yamada, 2004).

We are also interested in framing multi-class but *non*-scale-based categorization problems as metric labeling tasks. For example, positive vs. negative vs. neutral sentiment distinctions are sometimes considered in which neutral means either objective (Engström, 2004) or a conflation of objective with a rating of mediocre (Das and Chen, 2001). (Koppel and Schler (2005) in independent work also discuss various types of neutrality.) In either case, we could apply a metric in which positive and negative are closer to objective (or objective+mediocre) than to each other. As another example, hierarchical label relationships can be easily encoded in a label metric.

Finally, as mentioned in Section 3.3, we would like to address the transductive setting, in which one has a small amount of labeled data and uses relationships between unlabeled items, since it is particularly well-suited to the metric-labeling approach and may be quite important in practice.

Acknowledgments We thank Paul Bennett, Dave Blei, Claire Cardie, Shimon Edelman, Thorsten Joachims, Jon Kleinberg, Oren Kurland, John Lafferty, Guy Lebanon, Pradeep Ravikumar, Jerry Zhu, and the anonymous reviewers for many very useful comments and discussion. We learned of Moshe Koppel and Jonathan Schler's work while preparing the camera-ready version of this paper; we thank them for so quickly answering our request for a pre-print. Our descriptions of their work are based on that pre-print; we apologize in advance for any inaccuracies in our descriptions that result from changes between their pre-print and their final version. We also thank CMU for its hospitality during the year. This paper is based upon work supported in part by the National Science Foundation (NSF) under grant no. IIS-0329064 and CCR-0122581; SRI International under subcontract no. 03-000211 on their project funded by the Department of the Interior's National Business Center; and by an Alfred P. Sloan Research Fellowship. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of any sponsoring institutions, the U.S. government, or any other entity.

References

Atkeson, Christopher G., Andrew W. Moore, and Stefan Schaal. 1997. Locally weighted learning. *Artificial Intelligence Review*, 11(1):11–73.

Boykov, Yuri, Olga Veksler, and Ramin Zabih. 1999. Fast approximate energy minimization via graph cuts. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 377–384. Journal version in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 23(11):1222–1239, 2001.

Collins-Thompson, Kevyn and Jamie Callan. 2004. A language modeling approach to predicting reading difficulty. In *HLT-NAACL: Proceedings of the Main Conference*, pages 193–200.

Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific Finance Association Annual Conference (APFA)*.

Dave, Kushal, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of WWW*, pages 519–528.

Engström, Charlotta. 2004. Topic dependence in sentiment classification. Master's thesis, University of Cambridge.

Herbrich, Ralf, Thore Graepel, and Klaus Obermayer. 2000. Large margin rank boundaries for ordinal regression. In Alexander J. Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, Neural Information Processing Systems. MIT Press, pages 115–132.

Horvitz, Eric, Andy Jacobs, and David Hovel. 1999. Attention-sensitive alerting. In *Proceedings of the Conference on Uncertainty and Artificial Intelligence*, pages 305–313.

Ishikawa, Hiroshi. 2003. Exact optimization for Markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10).

Ishikawa, Hiroshi and Davi Geiger. 1998. Occlusions, discontinuities, and epipolar lines in stereo. In *Proceedings of the 5th European Conference on Computer Vision (ECCV)*, volume I, pages 232–248, London, UK. Springer-Verlag.

Joachims, Thorsten. 1999. Making large-scale SVM learning practical. In Bernhard Schölkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, pages 44–56.

Kleinberg, Jon and Éva Tardos. 2002. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *Journal of the ACM*, 49(5):616–639.

Koppel, Moshe and Jonathan Schler. 2005. The importance of neutral examples for learning sentiment. In *Workshop on the Analysis of Informal and Formal Information Exchange during Negotiations (FINEXIN)*.

Liu, Hugo, Henry Lieberman, and Ted Selker. 2003. A model of textual affect sensing using real-world knowledge. In *Proceedings of Intelligent User Interfaces (IUI)*, pages 125–132.

McCallum, Andrew. 1999. Multi-label text classification with a mixture model trained by EM. In *AAAI Workshop on Text Learning*.

McCullagh, Peter. 1980. Regression models for ordinal data. *Journal of the Royal Statistical Society*, 42(2):109–42.

Pang, Bo and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Qu, Yan, James Shanahan, and Janyce Wiebe, editors. 2004. *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*. AAAI Press. AAAI technical report SS-04-07.

Rifkin, Ryan M. and Aldebaro Klautau. 2004. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141.

Schapire, Robert E. and Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

Smola, Alex J. and Bernhard Schölkopf. 1998. A tutorial on support vector regression. Technical Report NeuroCOLT NC-TR-98-030, Royal Holloway College, University of London.

Subasic, Pero and Alison Huettner. 2001. Affect analysis of text using fuzzy semantic typing. *IEEE Transactions on Fuzzy Systems*, 9(4):483–496.

Takamura, Hiroya, Yuji Matsumoto, and Hiroyasu Yamada. 2004. Modeling category structures with a kernel function. In *Proceedings of CoNLL*, pages 57–64.

Tong, Richard M. 2001. An operational system for detecting and tracking opinions in on-line discussion. SIGIR Workshop on Operational Text Classification.

Turney, Peter. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the ACL*, pages 417–424.

Vapnik, Vladimir. 1995. *The Nature of Statistical Learning Theory*. Springer.

Wilson, Theresa, Janyce Wiebe, and Rebecca Hwa. 2004. Just how mad are you? Finding strong and weak opinion clauses. In *Proceedings of AAAI*, pages 761–769.

Yu, Hong and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP*.

Zhu, Xiaojin (Jerry). 2005. *Semi-Supervised Learning with Graphs*. Ph.D. thesis, Carnegie Mellon University.

A Appendix: other variations attempted

A.1 Discretizing binary classification

In our setting, we can also incorporate class relations by directly altering the output of a binary classifier, as follows. We first train a standard SVM, treating ratings greater than 0.5 as positive labels and others as negative labels. If we then consider the resulting classifier to output a *positivity-preference function* $\pi_+(x)$, we can then learn a series of thresholds to convert this value into the desired label set, under the assumption that the bigger $\pi_+(x)$ is, the more positive the review.[9] This algorithm always outperforms the majority-class baseline, but not to the degree that the best of SVM OVA and SVM regression does. Koppel and Schler (2005) independently found in a three-class study that thresholding a positive/negative classifier trained only on clearly positive or clearly negative examples did not yield large improvements.

A.2 Discretizing regression

In our experiments with SVM regression, we discretized regression output via a set of fixed decision thresholds $\{0.5, 1.5, 2.5, ...\}$ to map it into our set of class labels. Alternatively, we can learn the thresholds instead. Neither option clearly outperforms the other in the four-class case. In the three-class setting, the learned version provides noticeably better performance in two of the four datasets. But these results taken together still mean that in many cases, the difference is negligible, and if we had started down this path, we would have needed to consider similar tweaks for one-vs-all SVM as well. We therefore stuck with the simpler version in order to maintain focus on the central issues at hand.

[9]This is not necessarily true: if the classifier's goal is to optimize binary classification error, its major concern is to increase confidence in the positive/negative distinction, which may not correspond to higher confidence in separating "five stars" from "four stars".

Inducing Ontological Co-occurrence Vectors

Patrick Pantel
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
pantel@isi.edu

Abstract

In this paper, we present an unsupervised methodology for propagating lexical co-occurrence vectors into an ontology such as WordNet. We evaluate the framework on the task of automatically attaching new concepts into the ontology. Experimental results show 73.9% attachment accuracy in the first position and 81.3% accuracy in the top-5 positions. This framework could potentially serve as a foundation for ontologizing lexical-semantic resources and assist the development of other large-scale and internally consistent collections of semantic information.

1 Introduction

Despite considerable effort, there is still today no commonly accepted semantic corpus, semantic framework, notation, or even agreement on precisely which aspects of semantics are most useful (if at all). We believe that one important reason for this rather startling fact is the absence of truly wide-coverage semantic resources.

Recognizing this, some recent work on wide coverage term banks, like WordNet (Miller 1990) and CYC (Lenat 1995), and annotated corpora, like FrameNet (Baker et al. 1998), Propbank (Kingsbury et al. 2002) and Nombank (Meyers et al. 2004), seeks to address the problem. But manual efforts such as these suffer from two drawbacks: they are difficult to tailor to new domains, and they have internal inconsistencies that can make automating the acquisition process difficult.

In this work, we introduce a general framework for inducing co-occurrence feature vectors for nodes in a WordNet-like ontology. We believe that this framework will be useful for a variety of applications, including adding additional semantic information to existing semantic term banks by disambiguating lexical-semantic resources.

Ontologizing semantic resources

Recently, researchers have applied text- and web-mining algorithms for automatically creating lexical semantic resources like similarity lists (Lin 1998), semantic lexicons (Riloff and Shepherd 1997), hyponymy lists (Shinzato and Torisawa 2004; Pantel and Ravichandran 2004), part-whole lists (Girgu et al. 2003), and verb relation graphs (Chklovski and Pantel 2004). However, none of these resources have been directly linked into an ontological framework. For example, in VERBOCEAN (Chklovski and Pantel 2004), we find the verb relation "to surpass *is-stronger-than* to hit", but it is not specified that it is the achieving sense of *hit* where this relation applies.

We term *ontologizing* a lexical-semantic resource as the task of sense disambiguating the resource. This problem is different but not orthogonal to word-sense disambiguation. If we could disambiguate large collections of text with high accuracy, then current methods for building lexical-semantic resources could easily be applied to ontologize them by treating each word's senses as separate words. Our method does not require the disambiguation of text. Instead, it relies on the principle of distributional similarity and that polysemous words that are similar in one sense are dissimilar in their other senses.

Given the enriched ontologies produced by our method, we believe that ontologizing lexical-semantic resources will be feasible. For example, consider the example verb relation "to surpass *is-stronger-than* to hit" from above. To disambiguate the verb *hit*, we can look at all other verbs that *to surpass* is stronger than (for example, in VERBOCEAN, "to surpass *is-stronger-than* to overtake" and "to surpass *is-stronger-than* to equal"). Now, we can simply compare the lexical co-occurrence vectors of *overtake* and *equal* with the ontological feature vectors of the senses of *hit* (which are induced by our framework). The sense whose feature vector is most similar is selected.

It remains to be seen in future work how well this approach performs on ontologizing various semantic resources. In this paper, we focus on the general framework for inducing the ontological co-occurrence vectors and we apply it to the task of linking new terms into the ontology.

2 Relevant work

Our framework aims at enriching WordNet-like ontologies with syntactic features derived from a non-annotated corpus. Others have also made significant additions to WordNet. For example, in eXtended WordNet (Harabagiu et al. 1999), the rich glosses in WordNet are enriched by disambiguating the nouns, verbs, adverbs, and adjectives with synsets. Another work has enriched WordNet synsets with topically related words extracted from the Web (Agirre et al. 2001). While this method takes advantage of the redundancy of the web, our source of information is a local document collection, which opens the possibility for domain specific applications.

Distributional approaches to building semantic repositories have shown remarkable power. The underlying assumption, called the Distributional Hypothesis (Harris 1985), links the semantics of words to their lexical and syntactic behavior. The hypothesis states that words that occur in the same contexts tend to have similar meaning. Researchers have mostly looked at representing words by their surrounding words (Lund and Burgess 1996) and by their syntactical contexts (Hindle 1990; Lin 1998). However, these representations do not distinguish between the different senses of words. Our framework utilizes these principles and representations to induce disambiguated feature vectors. We describe these representations further in Section 3.

In supervised word sense disambiguation, senses are commonly represented by their surrounding words in a sense-tagged corpus (Gale et al. 1991). If we had a large collection of sense-tagged text, then we could extract disambiguated feature vectors by collecting co-occurrence features for each word sense. However, since there is little sense-tagged text available, the feature vectors for a random WordNet concept would be very sparse. In our framework, feature vectors are induced from much larger untagged corpora (currently 3GB of newspaper text).

Another approach to building semantic repositories is to collect and merge existing ontologies. Attempts to automate the merging process have not been particularly successful (Knight and Luk 1994; Hovy 1998; Noy and Musen 1999). The principal problems of partial and unbalanced coverage and of inconsistencies between ontologies continue to hamper these approaches.

3 Resources

The framework we present in Section 4 propagates any type of lexical feature up an ontology. In previous work, lexicals have often been represented by proximity and syntactic features. Consider the following sentence:

```
The tsunami left a trail of horror.
```

In a proximity approach, a word is represented by a window of words surrounding it. For the above sentence, a window of size 1 would yield two features (*-1:the* and *+1:left*) for the word *tsunami*. In a syntactic approach, more linguistically rich features are extracted by using each grammatical relation in which a word is involved (e.g. the features for *tsunami* are *determiner:the* and *subject-of:leave*).

For the purposes of this work, we consider the propagation of syntactic features. We used Minipar (Lin 1994), a broad coverage parser, to analyze text. We collected the statistics on the grammatical relations (contexts) output by Minipar and used these as the feature vectors. Following Lin (1998), we measure each feature f for a word e not by its frequency but by its pointwise mutual information, mi_{ef}:

Input: A node n and a corpus C.

Step 1: Termination Condition:
If n is a leaf node then assign to n its lexical feature vector as described in Section 3.

Step 2: Recursion Step:
For each child c of n, reecurse on c and C.
Assign a feature vector to n by propagating features from its children.

Output: A feature vector assigned to each node of the tree rooted by n.

Figure 1. Divide-and-conquer phase.

$$mi_{ef} = \log \frac{P(e, f)}{P(e) \times P(f)}$$

4 Inducing ontological features

The resource described in the previous section yields lexical feature vectors for each word in a corpus. We term these vectors *lexical* because they are collected by looking only at the lexicals in the text (i.e. no sense information is used). We use the term *ontological feature vector* to refer to a feature vector whose features are for a particular sense of the word.

In this section, we describe our framework for inducing ontological feature vectors for each node of an ontology. Our approach employs two phases. A divide-and-conquer algorithm first propagates syntactic features to each node in the ontology. A final sweep over the ontology, which we call the Coup phase, disambiguates the feature vectors of lexicals (leaf nodes) in the ontology.

4.1 Divide-and-conquer phase

In the first phase of the algorithm, we propagate features up the ontology in a bottom-up approach. Figure 1 gives an overview of this phase.

The termination condition of the recursion is met when the algorithm processes a leaf node. The feature vector that is assigned to this node is an exact copy of the lexical feature vector for that leaf (obtained from a large corpus as described in Section 3). For example, for the two leaf nodes labeled *chair* in Figure 2, we assign to both the same ambiguous lexical feature vector, an excerpt of which is shown in Figure 3.

When the recursion meets a non-leaf node, like *chairwoman* in Figure 2, the algorithm first recursively applies itself to each of the node's children. Then, the algorithm selects those features common to its children to propagate up to its own ontological feature vector. The assumption here is that features of other senses of polysemous words will not be propagated since they will not be common across the children. Below, we describe the two methods we used to propagate features: *Shared* and *Committee*.

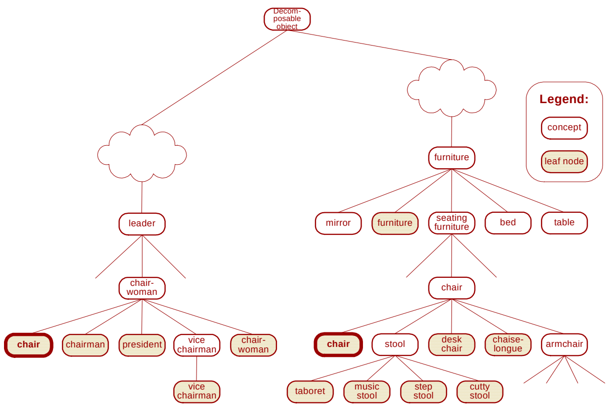


Figure 2. Subtrees of WordNet illustrating two senses of *chair*.

```
"chair"
   conjunction:
       sofa                 77     11.8
       professor            11     6.0
       dining room          2      5.6
       cushion              1      4.5
       council member       1      4.4
       President            9      2.9
       foreign minister     1      2.8
   nominal subject
       Ottoman              8      12.1
       director             22     9.1
       speaker              8      8.6
       Joyner               2      8.22
       recliner             2      7.7
       candidate            1      3.5
```

Figure 3. Excerpt of a lexical feature vector for the word *chair*. Grammatical relations are in italics (*conjunction* and *nominal-subject*). The first column of numbers are frequency counts and the other are mutual information scores. In bold are the features that intersect with the induced ontological feature vector for the parent concept of *chair*'s *chairwoman* sense.

Shared propagation algorithm

The first technique for propagating features to a concept node n from its children C is the simplest and scored best in our evaluation (see Section 5.2). The goal is that the feature vector for n

represents the general grammatical behavior that its children will have. For example, for the concept node *furniture* in Figure 2, we would like to assign features like *object-of:clean* since mosttypes of furniture can be cleaned. However, even though you can eat on a table, we do not want the feature *on:eat* for the *furniture* concept since we do not eat on *mirrors* or *beds*.

In the *Shared* propagation algorithm, we propagate only those features that are shared by at least *t* children. In our experiments, we experimentally set $t = min(3, |C|)$.

The frequency of a propagated feature is obtained by taking a weighted sum of the frequency of the feature across its children. Let f_i be the frequency of the feature for child *i*, let c_i be the total frequency of child *i*, and let *N* be the total frequency of all children. Then, the frequency *f* of the propagated feature is given by:

$$f = \sum_i f_i \times \frac{c_i}{N} \qquad (1)$$

Committee propagation algorithm

The second propagation algorithm finds a set of representative children from which to propagate features. Pantel and Lin (2002) describe an algorithm, called Clustering By Committee (CBC), which discovers clusters of words according to their meanings in test. The key to CBC is finding for each class a set of representative elements, called a committee, which most unambiguously describe the members of the class. For example, for the *color* concept, CBC discovers the following committee members:

```
purple, pink, yellow, mauve, turquoise,
beige, fuchsia
```

Words like *orange* and *violet* are avoided because they are polysemous. For a given concept *c*, we build a committee by clustering its children according to their similarity and then keep the largest and most interconnected cluster (see Pantel and Lin (2002) for details).

The propagated features are then those that are shared by at least two committee members. The frequency of a propagated feature is obtained using Eq. 1 where the children *i* are chosen only among the committee members.

Generating committees using CBC works best for classes with many members. In its original application (Pantel and Lin 2002), CBC discovered a flat list of coarse concepts. In the finer grained concept hierarchy of WordNet, there are many fewer children for each concept so we expect to have more difficulty finding committees.

Input:	A node *n* and the enriched ontology *O* output from the algorithm in Figure 1.
Step 1:	If *n* is not a leaf node then return.
Step 2:	Remove from *n*'s feature vector all features that intersect with the feature vector of any of *n*'s other senses' parent concepts, but are not in *n*'s parent concept feature vector.
Output:	A disambiguated feature vector for each leaf node *n*.

Figure 4. Coup phase.

4.2 Coup phase

At the end of the *Divide-and-conquer* phase, the non-leaf nodes of the ontology contain disambiguated features[1]. By design of the propagation algorithm, each concept node feature is shared by at least two of its children. We assume that two polysemous words, w_1 and w_2, that are similar in one sense will be dissimilar in its other senses. Under the distributional hypothesis, similar words occur in the same grammatical contexts and dissimilar words occur in different grammatical contexts. We expect then that most features that are shared between w_1 and w_2 will be the grammatical contexts of their similar sense. Hence, mostly disambiguated features are propagated up the ontology in the *Divide-and-conquer* phase.

However, the feature vectors for the leaf nodes remain ambiguous (e.g. the feature vectors for both leaf nodes labeled *chair* in Figure 2 are identical). In this phase of the algorithm, leaf node feature vectors are disambiguated by looking at the parents of their other senses.

Leaf nodes that are unambiguous in the ontology will have unambiguous feature vectors. For ambiguous leaf nodes (i.e. leaf nodes that have more than one concept parent), we apply the algorithm described in Figure 4. Given a polysemous leaf node *n*, we remove from its ambiguous

[1] By disambiguated features, we mean that the features are co-occurrences with a particular sense of a word; the features themselves are not sense-tagged.

feature vector those features that intersect with the ontological feature vector of any of its other senses' parent concept but that are not in its own parent's ontological feature vector. For example, consider the *furniture* sense of the leaf node *chair* in Figure 2. After the *Divide-and-conquer* phase, the node *chair* is assigned the ambiguous lexical feature vector shown in Figure 3. Suppose that *chair* only has one other sense in WordNet, which is the *chairwoman* sense illustrated in Figure 2. The features in bold in Figure 3 represent those features of *chair* that intersect with the ontological feature vector of *chairwoman*. In the *Coup* phase of our system, we remove these bold features from the *furniture* sense leaf node *chair*. What remains are features like "*chair and sofa*", "*chair and cushion*", "*Ottoman is a chair*", and "*recliner is a chair*". Similarly, for the *chairwoman* sense of *chair*, we remove those features that intersect with the ontological feature vector of the *chair* concept (the parent of the other *chair* leaf node).

As shown in the beginning of this section, concept node feature vectors are mostly unambiguous after the *Divide-and-conquer* phase. However, the *Divide-and-conquer* phase may be repeated after the *Coup* phase using a different termination condition. Instead of assigning to leaf nodes ambiguous lexical feature vectors, we use the leaf node feature vectors from the *Coup* phase. In our experiments, we did not see any significant performance difference by skipping this extra *Divide-and-conquer* step.

5 Experimental results

In this section, we provide a quantitative and qualitative evaluation of our framework.

5.1 Experimental Setup

We used Minipar (Lin 1994), a broad coverage parser, to parse two 3GB corpora (TREC-9 and TREC-2002). We collected the frequency counts of the grammatical relations (contexts) output by Minipar and used these to construct the lexical feature vectors as described in Section 3.

WordNet 2.0 served as our testing ontology. Using the algorithm presented in Section 4, we induced ontological feature vectors for the noun nodes in WordNet using the lexical co-occurrence features from the TREC-2002 corpus. Due to memory limitations, we were only able to propagate features to one quarter of the ontology. We experimented with both the *Shared* and *Committee* propagation models described in Section 4.1.

5.2 Quantitative evaluation

To evaluate the resulting ontological feature vectors, we considered the task of attaching new nodes into the ontology. To automatically evaluate this, we randomly extracted a set of 1000 noun leaf nodes from the ontology and accumulated lexical feature vectors for them using the TREC-9 corpus (a separate corpus than the one used to propagate features, but of the same genre). We experimented with two test sets:

- *Full*: The 424 of the 1000 random nodes that existed in the TREC-9 corpus
- *Subset*: Subset of *Full* where only nodes that do not have concept siblings are kept (380 nodes).

For each random node, we computed the similarity of the node with each concept node in the ontology by computing the cosine of the angle (Salton and McGill 1983) between the lexical feature vector of the random node e_i and the ontological feature vector of the concept nodes e_j:

$$sim(e_i, e_j) = \frac{\sum_f mi_{e_i f} \times mi_{e_j f}}{\sqrt{\sum_f mi_{e_i f}^2 \times \sum_f mi_{e_j f}^2}}$$

We only kept those similar nodes that had a similarity above a threshold σ. We experimentally set σ = 0.1.

Top-*K* accuracy

We collected the top-*K* most similar concept nodes (attachment points) for each node in the test sets and computed the accuracy of finding a correct attachment point in the top-*K* list. Table 1 shows the result.

We expected the algorithm to perform better on the *Subset* data set since only concepts that have exclusively lexical children must be considered for attachment. In the *Full* data set, the algorithm must consider each concept in the ontology as a potential attachment point. However, considering the top-5 best attachments, the algorithm performed equally well on both data sets.

The *Shared* propagation algorithm performed consistently slightly better than the *Committee* method. As described in Section 4.1, building a

Table 1. Correct attachment point in the top-*K* attachments (with 95% conf.)

K	*Shared (Full)*	*Committee (Full)*	*Shared (Subset)*	*Committee (Subset)*
1	73.9% ± 4.5%	72.0% ± 4.9%	77.4% ± 3.6%	76.1% ± 5.1%
2	78.7% ± 4.1%	76.6% ± 4.2%	80.7% ± 4.0%	79.1% ± 4.5%
3	79.9% ± 4.0%	78.2% ± 4.2%	81.2% ± 3.9%	80.5% ± 4.8%
4	80.6% ± 4.1%	79.0% ± 4.0%	81.5% ± 4.1%	80.8% ± 5.0%
5	81.3% ± 3.8%	79.5% ± 3.9%	81.7% ± 4.1%	81.3% ± 4.9%

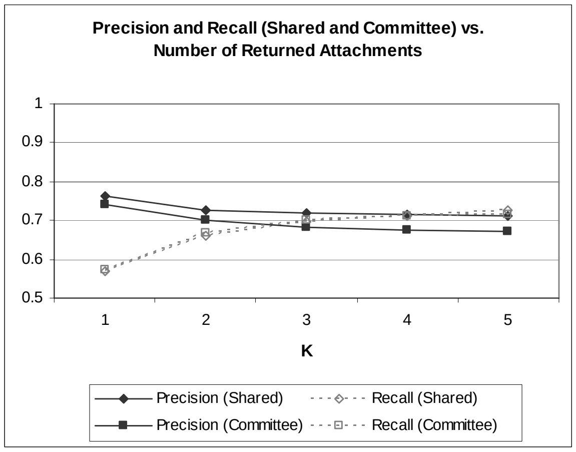


Figure 5. Attachment precision and recall for the *Shared* and *Committee* propagation methods when returning at most *K* attachments (on the *Full* set).

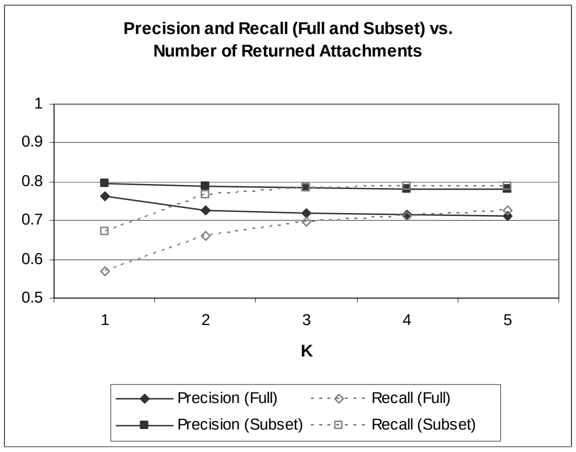


Figure 6. Attachment precision and recall for the *Full* and *Subset* data sets when returning at most *K* attachments (using the *Shared* propagation method).

committee performs best for concepts with many children. Since many nodes in WordNet have few direct children, the *Shared* propagation method is more appropriate. One possible extension of the *Committee* propagation algorithm is to find committee members from the full list of descendants of a node rather than only its immediate children.

Precision and Recall

We computed the precision and recall of our system on varying numbers of returned attachments. Figure 5 and Figure 6 show the attachment precision and recall of our system when the maximum number of returned attachments ranges between 1 and 5. In Figure 5, we see that the *Shared* propagation method has better precision than the *Committee* method. Both methods perform similarly on recall. The recall of the system increases most dramatically when returning two attachments without too much of a hit on precision. The low recall when returning only one attachment is due to both system errors and also to the fact that many nodes in the hierarchy are polysemous. In the next section, we discuss further experiments on polysemous nodes. Figure 6 illustrates the large difference on both precision and recall when using the simpler *Subset* data set. All 95% confidence bounds in Figure 5 and Figure 6 range between ±2.8% and ±5.3%.

Polysemous nodes

84 of the nodes in the Full data set are polysemous (they are attached to more than one concept node in the ontology). On average, these nodes have 2.6 senses for a total of 219 senses. Figure 7 compares the precision and recall of the system on all nodes in the *Full* data set vs. the 84 polysemous nodes. The 95% confidence intervals range between ±3.8% and ±5.0% for the *Full* data set and between ±1.2% and ±9.4% for the polysemous nodes. The precision on the polysemous nodes is consistently better since these have more possible correct attachments.

Clearly, when the system returns at most one or two attachments, the recall on the polysemous nodes is lower than on the *Full* set. However, it is interesting to note that recall on the polysemous nodes equals the recall on the *Full* set after *K*=3.

5.3 Qualitative evaluation

Inspection of errors revealed that the system often makes plausible attachments. Table 2 shows some example errors generated by our system. For the word *arsenic*, the system attached it to the concept *trioxide*, which is the parent of the correct attachment.

The system results may be useful to help validate the ontology. For example, for the word *law*, the system attached it to the *regulation* (as an organic process) and *ordinance* (legislative act) concepts. According to WordNet, *law* has seven possible attachment points, none of which are a legislative act. Hence, the system has found that in the TREC-9 corpus, the word *law* has a sense of *legislative act*. Similarly, the system discovered the *symptom* sense of *vomiting*.

The system discovered a potential anomaly in WordNet with the word *slob*. The system classified *slob* as follows:

```
fool → simpleton → someone
```

whereas WordNet classifies it as:

```
vulgarian → unpleasant person → unwelcome person → someone
```

The ontology could use this output to verify if *fool* should link in the *unpleasant person* subtree.

Capitalization is not very trustworthy in large collections of text. One of our design decisions was to ignore the case of words in our corpus, which in turn caused some errors since WordNet is case sensitive. For example, the lexical node *Munch* (Norwegian artist) was attached to the *munch* concept (food) by error because our system accumulated all features of the word *Munch* in text regardless of its capitalization.

6 Discussion

One question that remains unanswered is how clean an ontology must be in order for our methodology to work. Since the structure of the ontology guides the propagation of features, a very noisy ontology will result in noisy feature vectors. However, the framework is tolerant to some amount of noise and can in fact be used to correct some errors (as shown in Section 5.3).

We showed in Section 1 how our framework can be used to disambiguate lexical-semantic resources like hyponym lists, verb relations, and unknown words or terms. Other avenues of future work include:

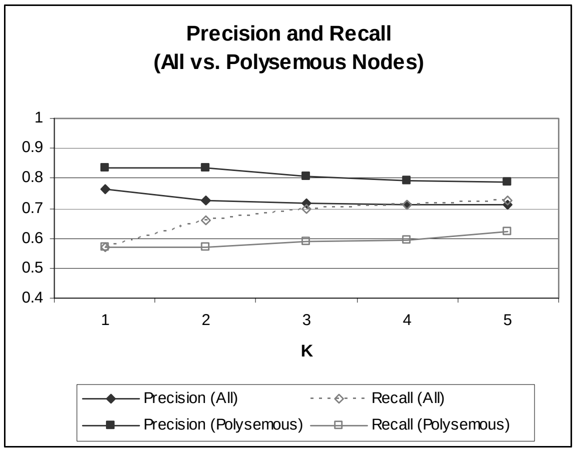


Figure 7. Attachment precision and recall on the *Full* set vs. the polysemous nodes in the *Full* set when the system returns at most *K* attachments.

Adapting/extending existing ontologies

It takes a large amount of time to build resources like WordNet. However, adapting existing resources to a new corpus might be possible using our framework. Once we have enriched the ontology with features from a corpus, we can rearrange the ontological structure according to the inter-conceptual similarity of nodes. For example, consider the word *computer* in WordNet, which has two senses: *a*) a machine; and *b*) a person who calculates. In a computer science corpus, sense *b*) occurs very infrequently and possibly a new sense of computer (e.g. *a processing chip*) occurs. A system could potentially remove sense *b*) since the similarity of the other children of *b*) and *computer* is very low. It could also uncover the new *processing chip* sense by finding a high similarity between *computer* and the *processing chip* concept.

Validating ontologies

This is a holy grail problem in the knowledge representation community. As a small step, our framework can be used to flag potential anomalies to the knowledge engineer.

What makes a *chair* different from a *recliner*?

Given an enriched ontology, we can remove from the feature vectors of *chair* and *recliner* those features that occur in their parent *furniture* concept. The features that remain describe their different syntactic behaviors in text.

Table 2. Example attachment errors by our system.

Node	*System Attachment*	*Correct Attachment*
arsenic*	trioxide	arsenic OR element
law	regulation	law OR police OR …
Munch†	munch	Munch
slob	fool	slob
vomiting	fever	emesis

* the system's attachment was a parent of the correct attachment.
† error due to case mix-up (our algorithm does not differentiate between case).

7 Conclusions

We presented a framework for inducing ontological feature vectors from lexical co-occurrence vectors. Our method does not require the disambiguation of text. Instead, it relies on the principle of distributional similarity and the fact that polysemous words that are similar in one sense tend to be dissimilar in their other senses. On the task of attaching new words to WordNet using our framework, our experiments showed that the first attachment has 73.9% accuracy and that a correct attachment is in the top-5 attachments with 81.3% accuracy.

We believe this work to be useful for a variety of applications. Not only can sense selection tasks such as word sense disambiguation, parsing, and semantic analysis benefit from our framework, but more inference-oriented tasks such as question answering and text summarization as well. We hope that this work will assist with the development of other large-scale and internally consistent collections of semantic information.

References

Agirre, E.; Ansa, O.; Martinez, D.; and Hovy, E. 2001. Enriching WordNet concepts with topic signatures. In *Proceedings of the NAACL workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations.* Pittsburgh, PA.

Baker, C.; Fillmore, C.; and Lowe, J. 1998. The Berkeley FrameNet project. In *Proceedings of COLING-ACL*. Montreal, Canada.

Chklovski, T., and Pantel, P. VERBOCEAN: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of EMNLP-2004*. pp. 33-40. Barcelona, Spain.

Gale, W.; Church, K.; and Yarowsky, D. 1992. A method for disambiguating word senses in a large corpus. *Computers and Humanities*, 26:415-439.

Girju, R.; Badulescu, A.; and Moldovan, D. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of HLT/NAACL-03*. pp. 80-87. Edmonton, Canada.

Harabagiu, S.; Miller, G.; and Moldovan, D. 1999. WordNet 2 - A Morphologically and Semantically Enhanced Resource. In *Proceedings of SIGLEX-99*. pp.1-8. University of Maryland.

Harris, Z. 1985. Distributional structure. In: Katz, J. J. (ed.) *The Philosophy of Linguistics*. New York: Oxford University Press. pp. 26-47.

Hovy, E. 1998. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proceedings LREC-98*. pp. 535-542. Granada, Spain.

Hindle, D. 1990. Noun classification from predicate-argument structures. In *Proceedings of ACL-90*. pp. 268-275. Pittsburgh, PA.

Kingsbury, P; Palmer, M.; and Marcus, M. 2002. Adding semantic annotation to the Penn TreeBank. In *Proceedings of HLT-2002*. San Diego, California.

Knight, K. and Luk, S. K. 1994. Building a large-scale knowledge base for machine translation. In *Proceedings of AAAI-1994*. Seattle, WA.

Lenat, D. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33-38.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-98*. pp. 768-774. Montreal, Canada.

Lin, D. 1994. Principar - an efficient, broad-coverage, principle-based parser. *Proceedings of COLING-94*. pp. 42-48. Kyoto, Japan.

Lund, K. and Burgess, C. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28:203-208.

Meyers, A.; Reeves, R.; Macleod, C.; Szekely, R.; Zielinska, V.; Young, B.; and Grishman, R. Annotating noun argument structure for NomBank. In *Proceedings of LREC-2004*. Lisbon, Portugal.

Miller, G. 1990. WordNet: An online lexical database. *International Journal of Lexicography*, 3(4).

Noy, N. F. and Musen, M. A. 1999. An algorithm for merging and aligning ontologies: Automation and tool support. In *Proceedings of the Workshop on Ontology Management (AAAI-99)*. Orlando, FL.

Pantel, P. and Lin, D. 2002. Discovering Word Senses from Text. In *Proceedings of SIGKDD-02*. pp. 613-619. Edmonton, Canada.

Riloff, E. and Shepherd, J. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of EMNLP-1997.*

Salton, G. and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

Shinzato, K. and Torisawa, K. 2004. Acquiring hyponymy relations from web documents. In *Proceedings of HLT-NAACL-2004*. pp. 73-80. Boston, MA.

Extracting Semantic Orientations of Words using Spin Model

Hiroya Takamura **Takashi Inui** **Manabu Okumura**
Precision and Intelligence Laboratory
Tokyo Institute of Technology
4259 Nagatsuta Midori-ku Yokohama, 226-8503 Japan
{takamura,oku}@pi.titech.ac.jp,
tinui@lr.pi.titech.ac.jp

Abstract

We propose a method for extracting semantic orientations of words: desirable or undesirable. Regarding semantic orientations as spins of electrons, we use the mean field approximation to compute the approximate probability function of the system instead of the intractable actual probability function. We also propose a criterion for parameter selection on the basis of magnetization. Given only a small number of seed words, the proposed method extracts semantic orientations with high accuracy in the experiments on English lexicon. The result is comparable to the best value ever reported.

1 Introduction

Identification of emotions (including opinions and attitudes) in text is an important task which has a variety of possible applications. For example, we can efficiently collect opinions on a new product from the internet, if opinions in bulletin boards are automatically identified. We will also be able to grasp people's attitudes in questionnaire, without actually reading all the responds.

An important resource in realizing such identification tasks is a list of words with semantic orientation: positive or negative (desirable or undesirable). Frequent appearance of positive words in a document implies that the writer of the document would have a positive attitude on the topic. The goal of this paper is to propose a method for automatically creating such a word list from glosses (i.e., definition or explanation sentences) in a dictionary, as well as from a thesaurus and a corpus. For this purpose, we use *spin model*, which is a model for a set of electrons with spins. Just as each electron has a direction of spin (up or down), each word has a semantic orientation (positive or negative). We therefore regard words as a set of electrons and apply the mean field approximation to compute the average orientation of each word. We also propose a criterion for parameter selection on the basis of magnetization, a notion in statistical physics. Magnetization indicates the global tendency of polarization.

We empirically show that the proposed method works well even with a small number of seed words.

2 Related Work

Turney and Littman (2003) proposed two algorithms for extraction of semantic orientations of words. To calculate the association strength of a word with positive (negative) seed words, they used the number of hits returned by a search engine, with a query consisting of the word and one of seed words (e.g., "*word* NEAR good", "*word* NEAR bad"). They regarded the difference of two association strengths as a measure of semantic orientation. They also proposed to use Latent Semantic Analysis to compute the association strength with seed words. An empirical evaluation was conducted on 3596 words extracted from General Inquirer (Stone et al., 1966).

Hatzivassiloglou and McKeown (1997) focused on conjunctive expressions such as "simple and well-received" and "simplistic but well-received", where the former pair of words tend to have the same semantic orientation, and the latter tend to have the opposite orientation. They first classify each conjunctive expression into the same-orientation class or the different-orientation class. They then use the classified expressions to cluster words into the positive class and the negative class. The experiments were conducted with the dataset that they created on their own. Evaluation was limited to adjectives.

Kobayashi et al. (2001) proposed a method for extracting semantic orientations of words with bootstrapping. The semantic orientation of a word is determined on the basis of its gloss, if any of their 52 hand-crafted rules is applicable to the sentence. Rules are applied iteratively in the bootstrapping framework. Although Kobayashi et al.'s work provided an accurate investigation on this task and inspired our work, it has drawbacks: low recall and language dependency. They reported that the semantic orientations of only 113 words are extracted with precision 84.1% (the low recall is due partly to their large set of seed words (1187 words)). The hand-crafted rules are only for Japanese.

Kamps et al. (2004) constructed a network by connecting each pair of synonymous words provided by WordNet (Fellbaum, 1998), and then used the shortest paths to two seed words "good" and "bad" to obtain the semantic orientation of a word. Limitations of their method are that a synonymy dictionary is required, that antonym relations cannot be incorporated into the model. Their evaluation is restricted to adjectives. The method proposed by Hu and Liu (2004) is quite similar to the shortest-path method. Hu and Liu's method iteratively determines the semantic orientations of the words neighboring any of the seed words and enlarges the seed word set in a bootstrapping manner.

Subjective words are often semantically oriented. Wiebe (2000) used a learning method to collect subjective adjectives from corpora. Riloff et al. (2003) focused on the collection of subjective nouns.

We later compare our method with Turney and Littman's method and Kamps et al.'s method.

The other pieces of research work mentioned above are related to ours, but their objectives are different from ours.

3 Spin Model and Mean Field Approximation

We give a brief introduction to the spin model and the mean field approximation, which are well-studied subjects both in the statistical mechanics and the machine learning communities (Geman and Geman, 1984; Inoue and Carlucci, 2001; Mackay, 2003).

A spin system is an array of N electrons, each of which has a spin with one of two values "$+1$ (up)" or "-1 (down)". Two electrons next to each other energetically tend to have the same spin. This model is called *the Ising spin model*, or simply *the spin model* (Chandler, 1987). The energy function of a spin system can be represented as

$$E(\mathbf{x}, W) = -\frac{1}{2}\sum_{ij} w_{ij} x_i x_j, \qquad (1)$$

where x_i and x_j ($\in \mathbf{x}$) are spins of electrons i and j, matrix $W = \{w_{ij}\}$ represents weights between two electrons.

In a spin system, the variable vector $\mathbf{x}$ follows the Boltzmann distribution :

$$P(\mathbf{x}|W) = \frac{\exp(-\beta E(\mathbf{x}, W))}{Z(W)}, \qquad (2)$$

where $Z(W) = \sum_{\mathbf{x}} \exp(-\beta E(\mathbf{x}, W))$ is the normalization factor, which is called *the partition function* and β is a constant called *the inverse-temperature*. As this distribution function suggests, a configuration with a higher energy value has a smaller probability.

Although we have a distribution function, computing various probability values is computationally difficult. The bottleneck is the evaluation of $Z(W)$, since there are 2^N configurations of spins in this system.

We therefore approximate $P(\mathbf{x}|W)$ with a simple function $Q(\mathbf{x}; \theta)$. The set of parameters θ for Q, is determined such that $Q(\mathbf{x}; \theta)$ becomes as similar to $P(\mathbf{x}|W)$ as possible. As a measure for the distance between P and Q, the variational free energy F is often used, which is defined as the difference between the mean energy with respect to Q and the entropy of Q :

$$F(\theta) = \beta \sum_{\mathbf{x}} Q(\mathbf{x}; \theta) E(\mathbf{x}; W)$$

$$-\Big(-\sum_{\mathbf{x}} Q(\mathbf{x};\theta)\log Q(\mathbf{x};\theta)\Big). \quad (3)$$

The parameters θ that minimizes the variational free energy will be chosen. It has been shown that minimizing F is equivalent to minimizing the Kullback-Leibler divergence between P and Q (Mackay, 2003).

We next assume that the function $Q(\mathbf{x};\theta)$ has the factorial form :

$$Q(\mathbf{x};\theta) = \prod_i Q(x_i;\theta_i). \quad (4)$$

Simple substitution and transformation leads us to the following variational free energy :

$$F(\theta) = -\frac{\beta}{2}\sum_{ij} w_{ij}\bar{x}_i\bar{x}_j - \sum_i\Big(-\sum_{x_i} Q(x_i;\theta_i)\log Q(x_i;\theta_i)\Big). \quad (5)$$

With the usual method of Lagrange multipliers, we obtain the *mean field equation* :

$$\bar{x}_i = \frac{\sum_{x_i} x_i \exp\Big(\beta x_i \sum_j w_{ij}\bar{x}_j\Big)}{\sum_{x_i}\exp\Big(\beta x_i\sum_j w_{ij}\bar{x}_j\Big)}. \quad (6)$$

This equation is solved by the iterative update rule :

$$\bar{x}_i^{new} = \frac{\sum_{x_i} x_i \exp\Big(\beta x_i \sum_j w_{ij}\bar{x}_j^{old}\Big)}{\sum_{x_i}\exp\Big(\beta x_i\sum_j w_{ij}\bar{x}_j^{old}\Big)}. \quad (7)$$

4 Extraction of Semantic Orientation of Words with Spin Model

We use the spin model to extract semantic orientations of words.

Each spin has a direction taking one of two values: up or down. Two neighboring spins tend to have the same direction from a energetic reason. Regarding each word as an electron and its semantic orientation as the spin of the electron, we construct a lexical network by connecting two words if, for example, one word appears in the gloss of the other word. Intuition behind this is that if a word is semantically oriented in one direction, then the words in its gloss tend to be oriented in the same direction.

Using the mean-field method developed in statistical mechanics, we determine the semantic orientations on the network in a global manner. The global optimization enables the incorporation of possibly noisy resources such as glosses and corpora, while existing simple methods such as the shortest-path method and the bootstrapping method cannot work in the presence of such noisy evidences. Those methods depend on less-noisy data such as a thesaurus.

4.1 Construction of Lexical Networks

We construct a lexical network by linking two words if one word appears in the gloss of the other word. Each link belongs to one of two groups: the same-orientation links SL and the different-orientation links DL. If at least one word precedes a negation word (e.g., not) in the gloss of the other word, the link is a different-orientation link. Otherwise the links is a same-orientation link.

We next set weights $W = (w_{ij})$ to links :

$$w_{ij} = \begin{cases} \frac{1}{\sqrt{d(i)d(j)}} & (l_{ij} \in SL) \\ -\frac{1}{\sqrt{d(i)d(j)}} & (l_{ij} \in DL) \\ 0 & otherwise \end{cases}, \quad (8)$$

where l_{ij} denotes the link between word i and word j, and $d(i)$ denotes the degree of word i, which means the number of words linked with word i. Two words without connections are regarded as being connected by a link of weight 0. We call this network *the gloss network (G)*.

We construct another network, *the gloss-thesaurus network (GT)*, by linking synonyms, antonyms and hypernyms, in addition to the the above linked words. Only antonym links are in DL.

We enhance the gloss-thesaurus network with cooccurrence information extracted from corpus. As mentioned in Section 2, Hatzivassiloglou and McKeown (1997) used conjunctive expressions in corpus. Following their method, we connect two adjectives if the adjectives appear in a conjunctive form in the corpus. If the adjectives are connected by “and”, the link belongs to SL. If they are connected by “but”, the link belongs to DL. We call this network *the gloss-thesaurus-corpus network (GTC)*.

4.2 Extraction of Orientations

We suppose that a small number of seed words are given. In other words, we know beforehand the semantic orientations of those given words. We incorporate this small labeled dataset by modifying the previous update rule.

Instead of $\beta E(\mathbf{x}, W)$ in Equation (2), we use the following function $H(\beta, \mathbf{x}, W)$:

$$H(\beta, \mathbf{x}, W) = -\frac{\beta}{2}\sum_{ij} w_{ij}x_i x_j + \alpha\sum_{i\in L}(x_i - a_i)^2, \tag{9}$$

where L is the set of seed words, a_i is the orientation of seed word i, and α is a positive constant. This expression means that if x_i $(i \in L)$ is different from a_i, the state is penalized.

Using function H, we obtain the new update rule for x_i $(i \in L)$:

$$\bar{x}_i^{new} = \frac{\sum_{x_i} x_i \exp\Big(\beta x_i s_i^{old} - \alpha(x_i - a_i)^2\Big)}{\sum_{x_i} \exp\Big(\beta x_i s_i^{old} - \alpha(x_i - a_i)^2\Big)}, \tag{10}$$

where $s_i^{old} = \sum_j w_{ij}\bar{x}_j^{old}$. $\bar{x}_i^{old}$ and $\bar{x}_i^{new}$ are the averages of x_i respectively before and after update. What is discussed here was constructed with the reference to work by Inoue and Carlucci (2001), in which they applied the spin glass model to image restoration.

Initially, the averages of the seed words are set according to their given orientations. The other averages are set to 0.

When the difference in the value of the variational free energy is smaller than a threshold before and after update, we regard computation converged.

The words with high final average values are classified as positive words. The words with low final average values are classified as negative words.

4.3 Hyper-parameter Prediction

The performance of the proposed method largely depends on the value of hyper-parameter β. In order to make the method more practical, we propose criteria for determining its value.

When a large labeled dataset is available, we can obtain a reliable *pseudo leave-one-out error rate* :

$$\frac{1}{|L|}\sum_{i\in L}[a_i\bar{x}'_i], \tag{11}$$

where $[t]$ is 1 if t is negative, otherwise 0, and $\bar{x}'_i$ is calculated with the right-hand-side of Equation (6), where the penalty term $\alpha(\bar{x}_i - a_i)^2$ in Equation (10) is ignored. We choose β that minimizes this value.

However, when a large amount of labeled data is unavailable, the value of pseudo leave-one-out error rate is not reliable. In such cases, we use *magnetization* m for hyper-parameter prediction :

$$m = \frac{1}{N}\sum_i \bar{x}_i. \tag{12}$$

At a high temperature, spins are randomly oriented (*paramagnetic phase*, $m \approx 0$). At a low temperature, most of the spins have the same direction (*ferromagnetic phase*, $m \neq 0$). It is known that at some intermediate temperature, ferromagnetic phase suddenly changes to paramagnetic phase. This phenomenon is called *phase transition*. Slightly before the phase transition, spins are locally polarized; strongly connected spins have the same polarity, but not in a global way.

Intuitively, the state of the lexical network is locally polarized. Therefore, we calculate values of m with several different values of β and select the value just before the phase transition.

4.4 Discussion on the Model

In our model, the semantic orientations of words are determined according to the averages values of the spins. Despite the heuristic flavor of this decision rule, it has a theoretical background related to maximizer of posterior marginal (MPM) estimation, or ‘finite-temperature decoding’ (Iba, 1999; Marroquin, 1985). In MPM, the average is the marginal distribution over x_i obtained from the distribution over **x**. We should note that the finite-temperature decoding is quite different from annealing type algorithms or ‘zero-temperature decoding’, which correspond to maximum a posteriori (MAP) estimation and also often used in natural language processing (Cowie et al., 1992).

Since the model estimation has been reduced to simple update calculations, the proposed model is similar to conventional spreading activation approaches, which have been applied, for example, to word sense disambiguation (Veronis and Ide, 1990). Actually, the proposed model can be regarded as a spreading activation model with a specific update rule, as long as we are dealing with 2-class model (2-Ising model).

However, there are some advantages in our modelling. The largest advantage is its theoretical background. We have an objective function and its approximation method. We thus have a measure of goodness in model estimation and can use another better approximation method, such as Bethe approximation (Tanaka et al., 2003). The theory tells us which update rule to use. We also have a notion of magnetization, which can be used for hyper-parameter estimation. We can use a plenty of knowledge, methods and algorithms developed in the field of statistical mechanics. We can also extend our model to a multiclass model (*Q-Ising model*).

Another interesting point is the relation to maximum entropy model (Berger et al., 1996), which is popular in the natural language processing community. Our model can be obtained by maximizing the entropy of the probability distribution $Q(\mathbf{x})$ under constraints regarding the energy function.

5 Experiments

We used glosses, synonyms, antonyms and hypernyms of WordNet (Fellbaum, 1998) to construct an English lexical network. For part-of-speech tagging and lemmatization of glosses, we used Tree-Tagger (Schmid, 1994). 35 stopwords (quite frequent words such as “be” and “have”) are removed from the lexical network. Negation words include 33 words. In addition to usual negation words such as “not” and “never”, we include words and phrases which mean negation in a general sense, such as “free from” and “lack of”. The whole network consists of approximately 88,000 words. We collected 804 conjunctive expressions from Wall Street Journal and Brown corpus as described in Section 4.2.

The labeled dataset used as a gold standard is General Inquirer lexicon (Stone et al., 1966) as in the work by Turney and Littman (2003). We extracted the words tagged with “Positiv” or “Negativ”, and reduced multiple-entry words to single entries. As a result, we obtained 3596 words (1616 positive words and 1980 negative words) [1]. In the computation of accuracy, seed words are eliminated from these 3596 words.

[1]Although we preprocessed in the same way as Turney and Littman, there is a slight difference between their dataset and our dataset. However, we believe this difference is insignificant.

Table 1: Classification accuracy (%) with various networks and four different sets of seed words. In the parentheses, the predicted value of β is written. For cv, no value is written for β, since 10 different values are obtained.

seeds	GTC	GT	G
cv	90.8 (—)	90.9 (—)	86.9 (—)
14	81.9 (1.0)	80.2 (1.0)	76.2 (1.0)
4	73.8 (0.9)	73.7 (1.0)	65.2 (0.9)
2	74.6 (1.0)	61.8 (1.0)	65.7 (1.0)

We conducted experiments with different values of β from 0.1 to 2.0, with the interval 0.1, and predicted the best value as explained in Section 4.3. The threshold of the magnetization for hyper-parameter estimation is set to 1.0×10^{-5}. That is, the predicted optimal value of β is the largest β whose corresponding magnetization does not exceeds the threshold value.

We performed 10-fold cross validation as well as experiments with fixed seed words. The fixed seed words are the ones used by Turney and Littman: 14 seed words {good, nice, excellent, positive, fortunate, correct, superior, bad, nasty, poor, negative, unfortunate, wrong, inferior}; 4 seed words {good, superior, bad, inferior}; 2 seed words {good, bad}.

5.1 Classification Accuracy

Table 1 shows the accuracy values of semantic orientation classification for four different sets of seed words and various networks. In the table, cv corresponds to the result of 10-fold cross validation, in which case we use the pseudo leave-one-out error for hyper-parameter estimation, while in other cases we use magnetization.

In most cases, the synonyms and the cooccurrence information from corpus improve accuracy. The only exception is the case of 2 seed words, in which G performs better than GT. One possible reason of this inversion is that the computation is trapped in a local optimum, since a small number of seed words leave a relatively large degree of freedom in the solution space, resulting in more local optimal points.

We compare our results with Turney and

Table 2: Actual best classification accuracy (%) with various networks and four different sets of seed words. In the parenthesis, the actual best value of β is written, except for cv.

seeds	GTC	GT	G
cv	91.5 (—)	91.5 (—)	87.0 (—)
14	81.9 (1.0)	80.2 (1.0)	76.2 (1.0)
4	74.4 (0.6)	74.4 (0.6)	65.3 (0.8)
2	75.2 (0.8)	61.9 (0.8)	67.5 (0.5)

Littman's results. With 14 seed words, they achieved 61.26% for a small corpus (approx. 1×10^7 words), 76.06% for a medium-sized corpus (approx. 2×10^9 words), 82.84% for a large corpus (approx. 1×10^{11} words).

Without a corpus nor a thesaurus (but with glosses in a dictionary), we obtained accuracy that is comparable to Turney and Littman's with a medium-sized corpus. When we enhance the lexical network with corpus and thesaurus, our result is comparable to Turney and Littman's with a large corpus.

5.2 Prediction of β

We examine how accurately our prediction method for β works by comparing Table 1 above and Table 2 below. Our method predicts good β quite well especially for 14 seed words. For small numbers of seed words, our method using magnetization tends to predict a little larger value.

We also display the figure of magnetization and accuracy in Figure 1. We can see that the sharp change of magnetization occurs at around $\beta = 1.0$ (phrase transition). At almost the same point, the classification accuracy reaches the peak.

5.3 Precision for the Words with High Confidence

We next evaluate the proposed method in terms of precision for the words that are classified with high confidence. We regard the absolute value of each average as a confidence measure and evaluate the top words with the highest absolute values of averages.

The result of this experiment is shown in Figure 2, for 14 seed words as an example. The top 1000 words achieved more than 92% accuracy. This result shows that the absolute value of each average

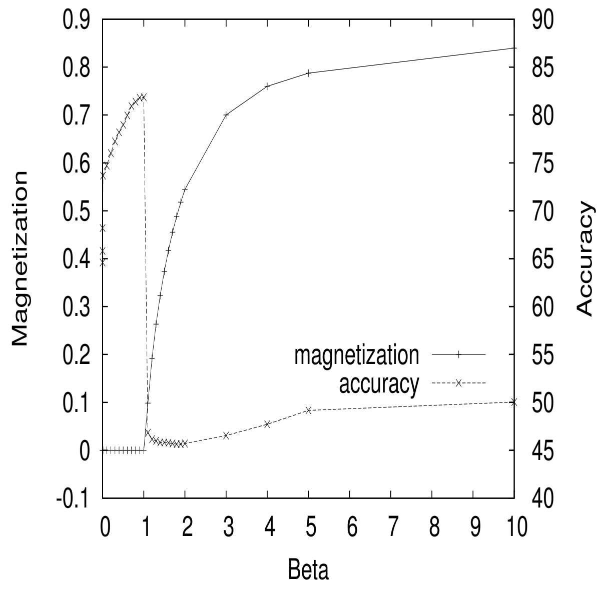


Figure 1: Example of magnetization and classification accuracy(14 seed words).

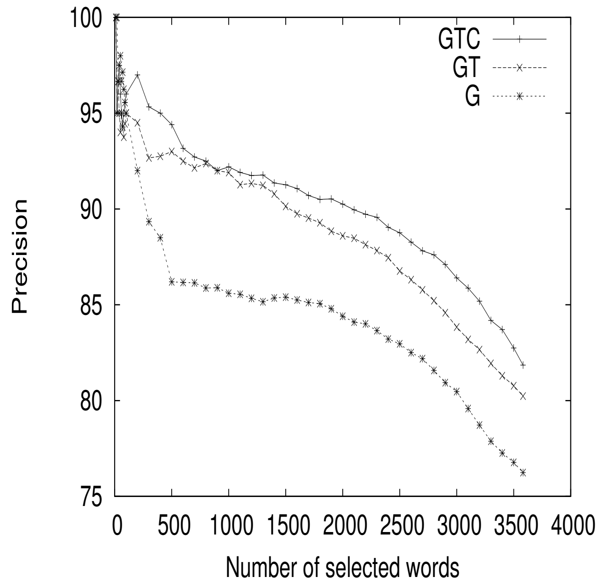


Figure 2: Precision (%) with 14 seed words.

Table 3: Precision (%) for selected adjectives. Comparison between the proposed method and the shortest-path method.

seeds	proposed	short. path
14	73.4 (1.0)	70.8
4	71.0 (1.0)	64.9
2	68.2 (1.0)	66.0

Table 4: Precision (%) for adjectives. Comparison between the proposed method and the bootstrapping method.

seeds	proposed	bootstrap
14	83.6 (0.8)	72.8
4	82.3 (0.9)	73.2
2	83.5 (0.7)	71.1

can work as a confidence measure of classification.

5.4 Comparison with other methods

In order to further investigate the model, we conduct experiments in restricted settings.

We first construct a lexical network using only synonyms. We compare the spin model with the shortest-path method proposed by Kamps et al. (2004) on this network, because the shortest-path method cannot incorporate negative links of antonyms. We also restrict the test data to 697 adjectives, which is the number of examples that the shortest-path method can assign a non-zero orientation value. Since the shortest-path method is designed for 2 seed words, the method is extended to use the average shortest-path lengths for 4 seed words and 14 seed words. Table 3 shows the result. Since the only difference is their algorithms, we can conclude that the global optimization of the spin model works well for the semantic orientation extraction.

We next compare the proposed method with a simple bootstrapping method proposed by Hu and Liu (2004). We construct a lexical network using synonyms and antonyms. We restrict the test data to 1470 adjectives for comparison of methods. The result in Table 4 also shows that the global optimization of the spin model works well for the semantic orientation extraction.

We also tested the shortest path method and the bootstrapping method on GTC and GT, and obtained low accuracies as expected in the discussion in Section 4.

5.5 Error Analysis

We investigated a number of errors and concluded that there were mainly three types of errors.

One is the ambiguity of word senses. For example, one of the glosses of “costly”is “entailing great loss or sacrifice”. The word “great” here means “large”, although it usually means “outstanding” and is positively oriented.

Another is lack of structural information. For example, “arrogance” means “overbearing pride evidenced by a superior manner toward the weak”. Although “arrogance” is mistakingly predicted as positive due to the word “superior”, what is superior here is “manner”.

The last one is idiomatic expressions. For example, although “brag” means “show off”, neither of “show” and “off” has the negative orientation. Idiomatic expressions often does not inherit the semantic orientation from or to the words in the gloss.

The current model cannot deal with these types of errors. We leave their solutions as future work.

6 Conclusion and Future Work

We proposed a method for extracting semantic orientations of words. In the proposed method, we regarded semantic orientations as spins of electrons, and used the mean field approximation to compute the approximate probability function of the system instead of the intractable actual probability function. We succeeded in extracting semantic orientations with high accuracy, even when only a small number of seed words are available.

There are a number of directions for future work.

One is the incorporation of syntactic information. Since the importance of each word consisting a gloss depends on its syntactic role. syntactic information in glosses should be useful for classification.

Another is active learning. To decrease the amount of manual tagging for seed words, an active learning scheme is desired, in which a small number of *good* seed words are automatically selected.

Although our model can easily extended to a multi-state model, the effectiveness of using such a multi-state model has not been shown yet.

Our model uses only the tendency of having the same orientation. Therefore we can extract semantic orientations of new words that are not listed in a dictionary. The validation of such extension will widen the possibility of application of our method.

Larger corpora such as web data will improve performance. The combination of our method and the method by Turney and Littman (2003) is promising.

Finally, we believe that the proposed model is applicable to other tasks in computational linguistics.

References

Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

David Chandler. 1987. *Introduction to Modern Statistical Mechanics*. Oxford University Press.

Jim Cowie, Joe Guthrie, and Louise Guthrie. 1992. Lexical disambiguation using simulated annealing. In *Proceedings of the 14th conference on Computational linguistics*, volume 1, pages 359–365.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database, Language, Speech, and Communication Series*. MIT Press.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and the Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 174–181.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining (KDD-2004)*, pages 168–177.

Yukito Iba. 1999. The nishimori line and bayesian statistics. *Journal of Physics A: Mathematical and General*, pages 3875–3888.

Junichi Inoue and Domenico M. Carlucci. 2001. Image restoration using the q-ising spin glass. *Physical Review E*, 64:036121–1 – 036121–18.

Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten de Rijke. 2004. Using wordnet to measure semantic orientation of adjectives. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004), volume IV*, pages 1115–1118.

Nozomi Kobayashi, Takashi Inui, and Kentaro Inui. 2001. Dictionary-based acquisition of the lexical knowledge for p/n analysis (in Japanese). In *Proceedings of Japanese Society for Artificial Intelligence, SLUD-33*, pages 45–50.

David J. C. Mackay. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

Jose L. Marroquin. 1985. Optimal bayesian estimators for image segmentation and surface reconstruction. Technical Report A.I. Memo 839, Massachusetts Institute of Technology.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-03)*, pages 25–32.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pages 44–49.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press.

Kazuyuki Tanaka, Junichi Inoue, and Mike Titterington. 2003. Probabilistic image processing by means of the bethe approximation for the q-ising model. *Journal of Physics A: Mathematical and General*, 36:11023–11035.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346.

Jean Veronis and Nancy M. Ide. 1990. Word sense disambiguation with very large neural networks extracted from machine readable dictionaries. In *Proceedings of the 13th Conference on Computational Linguistics*, volume 2, pages 389–394.

Janyce M. Wiebe. 2000. Learning subjective adjectives from corpora. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, pages 735–740.

Modeling Local Coherence: An Entity-based Approach

Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
regina@csail.mit.edu

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

Abstract

This paper considers the problem of automatic assessment of local coherence. We present a novel entity-based representation of discourse which is inspired by Centering Theory and can be computed automatically from raw text. We view coherence assessment as a ranking learning problem and show that the proposed discourse representation supports the effective learning of a ranking function. Our experiments demonstrate that the induced model achieves significantly higher accuracy than a state-of-the-art coherence model.

1 Introduction

A key requirement for any system that produces text is the coherence of its output. Not surprisingly, a variety of coherence theories have been developed over the years (e.g., Mann and Thomson, 1988; Grosz et al. 1995) and their principles have found application in many symbolic text generation systems (e.g., Scott and de Souza, 1990; Kibble and Power, 2004). The ability of these systems to generate high quality text, almost indistinguishable from human writing, makes the incorporation of coherence theories in robust large-scale systems particularly appealing. The task is, however, challenging considering that most previous efforts have relied on handcrafted rules, valid only for limited domains, with no guarantee of scalability or portability (Reiter and Dale, 2000). Furthermore, coherence constraints are often embedded in complex representations (e.g., Asher and Lascarides, 2003) which are hard to implement in a robust application.

This paper focuses on *local coherence*, which captures text relatedness at the level of sentence-to-sentence transitions, and is essential for generating *globally* coherent text. The key premise of our work is that the distribution of entities in locally coherent texts exhibits certain regularities. This assumption is not arbitrary — some of these regularities have been recognized in Centering Theory (Grosz et al., 1995) and other entity-based theories of discourse.

The algorithm introduced in the paper automatically abstracts a text into a set of entity transition sequences, a representation that reflects distributional, syntactic, and referential information about discourse entities. We argue that this representation of discourse allows the system to learn the properties of locally coherent texts opportunistically from a given corpus, without recourse to manual annotation or a predefined knowledge base.

We view coherence assessment as a ranking problem and present an efficiently learnable model that orders alternative renderings of the same information based on their degree of local coherence. Such a mechanism is particularly appropriate for generation and summarization systems as they can produce multiple text realizations of the same underlying content, either by varying parameter values, or by relaxing constraints that control the generation process. A system equipped with a ranking mechanism, could compare the quality of the candidate outputs, much in the same way speech recognizers employ *language models* at the sentence level.

Our evaluation results demonstrate the effectiveness of our entity-based ranking model within the general framework of coherence assessment. First, we evaluate the utility of the model in a text ordering task where our algorithm has to select a maximally coherent sentence order from a set of candidate permutations. Second, we compare the rankings produced by the model against human coherence judgments elicited for automatically generated summaries. In both experiments, our method yields a significant improvement over a state-of-the-art coherence model based on Latent Semantic Analysis (Foltz et al., 1998).

In the following section, we provide an overview of existing work on the automatic assessment of local coherence. Then, we introduce our entity-based representation, and describe our ranking model. Next, we present the experimental framework and data. Evaluation results conclude the paper.

2 Related Work

Local coherence has been extensively studied within the modeling framework put forward by Centering Theory (Grosz et al., 1995; Walker et al., 1998; Strube and Hahn, 1999; Poesio et al., 2004; Kibble and Power, 2004). One of the main assumptions underlying Centering is that a text segment which foregrounds a single entity is perceived to be more coherent than a segment in which multiple entities are discussed. The theory formalizes this intuition by introducing constraints on the distribution of discourse entities in coherent text. These constraints are formulated in terms of *focus*, the most salient entity in a discourse segment, and *transition* of focus between adjacent sentences. The theory also establishes constraints on the linguistic realization of focus, suggesting that it is more likely to appear in prominent syntactic positions (such as subject or object), and to be referred to with anaphoric expressions.

A great deal of research has attempted to translate principles of Centering Theory into a robust coherence metric (Miltsakaki and Kukich, 2000; Hasler, 2004; Karamanis et al., 2004). Such a translation is challenging in several respects: one has to specify the "free parameters" of the system (Poesio et al., 2004) and to determine ways of combining the effects of various constraints. A common methodology that has emerged in this research is to develop and evaluate coherence metrics on manually annotated corpora. For instance, Miltsakaki and Kukich (2000) annotate a corpus of student essays with transition information, and show that the distribution of transitions correlates with human grades. Karamanis et al. (2004) use a similar methodology to compare coherence metrics with respect to their usefulness for text planning in generation.

The present work differs from these approaches in two key respects. First, our method does not require manual annotation of input texts. We do not aim to produce complete centering annotations; instead, our inference procedure is based on a discourse representation that preserves essential entity transition information, and can be computed automatically from raw text. Second, we learn patterns of entity distribution from a corpus, without attempting to directly implement or refine Centering constraints.

3 The Coherence Model

In this section we introduce our entity-based representation of discourse. We describe how it can be computed and how entity transition patterns can be extracted. The latter constitute a rich feature space on which probabilistic inference is performed.

Text Representation Each text is represented by an *entity grid*, a two-dimensional array that captures the distribution of discourse entities across text sentences. We follow Miltsakaki and Kukich (2000) in assuming that our unit of analysis is the traditional sentence (i.e., a main clause with accompanying subordinate and adjunct clauses). The rows of the grid correspond to sentences, while the columns correspond to discourse entities. By *discourse entity* we mean a class of coreferent noun phrases. For each occurrence of a discourse entity in the text, the corresponding grid cell contains information about its grammatical role in the given sentence. Each grid column thus corresponds to a string from a set of categories reflecting the entity's presence or absence in a sequence of sentences. Our set consists of four symbols: **S** (subject), **O** (object), **X** (neither subject nor object) and **–** (gap which signals the entity's absence from a given sentence).

Table 1 illustrates a fragment of an entity grid constructed for the text in Table 2. Since the text contains six sentences, the grid columns are of length six. Consider for instance the grid column for the entity *trial*, **[O – – – – X]**. It records that *trial* is present in sentences 1 and 6 (as **O** and **X** respectively) but is absent from the rest of the sentences.

Grid Computation The ability to identify and cluster coreferent discourse entities is an important prerequisite for computing entity grids. The same entity may appear in different linguistic forms, e.g., *Microsoft Corp.*, *Microsoft*, and *the company*, but should still be mapped to a single entry in the grid. Table 1 exemplifies the entity grid for the text in Table 2 when coreference resolution is taken into account. To automatically compute entity classes,

	Department	Trial	Microsoft	Evidence	Competitors	Markets	Products	Brands	Case	Netscape	Software	Tactics	Government	Suit	Earnings	
1	S	O	S	X	O	–	–	–	–	–	–	–	–	–	–	1
2	–	–	O	–	–	X	S	O	–	–	–	–	–	–	–	2
3	–	–	S	O	–	–	–	–	S	O	O	–	–	–	–	3
4	–	–	S	–	–	–	–	–	–	–	–	S	–	–	–	4
5	–	–	–	–	–	–	–	–	–	–	–	–	S	O	–	5
6	–	X	S	–	–	–	–	–	–	–	–	–	–	–	O	6

Table 1: A fragment of the entity grid. Noun phrases are represented by their head nouns.

1	[The Justice Department]$_{\mathbf{S}}$ is conducting an [anti-trust trial]$_{\mathbf{O}}$ against [Microsoft Corp.]$_{\mathbf{X}}$ with [evidence]$_{\mathbf{X}}$ that [the company]$_{\mathbf{S}}$ is increasingly attempting to crush [competitors]$_{\mathbf{O}}$.
2	[Microsoft]$_{\mathbf{O}}$ is accused of trying to forcefully buy into [markets]$_{\mathbf{X}}$ where [its own products]$_{\mathbf{S}}$ are not competitive enough to unseat [established brands]$_{\mathbf{O}}$.
3	[The case]$_{\mathbf{S}}$ revolves around [evidence]$_{\mathbf{O}}$ of [Microsoft]$_{\mathbf{S}}$ aggressively pressuring [Netscape]$_{\mathbf{O}}$ into merging [browser software]$_{\mathbf{O}}$.
4	[Microsoft]$_{\mathbf{S}}$ claims [its tactics]$_{\mathbf{S}}$ are commonplace and good economically.
5	[The government]$_{\mathbf{S}}$ may file [a civil suit]$_{\mathbf{O}}$ ruling that [conspiracy]$_{\mathbf{S}}$ to curb [competition]$_{\mathbf{O}}$ through [collusion]$_{\mathbf{X}}$ is [a violation of the Sherman Act]$_{\mathbf{O}}$.
6	[Microsoft]$_{\mathbf{S}}$ continues to show [increased earnings]$_{\mathbf{O}}$ despite [the trial]$_{\mathbf{X}}$.

Table 2: Summary augmented with syntactic annotations for grid computation.

we employ a state-of-the-art noun phrase coreference resolution system (Ng and Cardie, 2002) trained on the MUC (6–7) data sets. The system decides whether two NPs are coreferent by exploiting a wealth of features that fall broadly into four categories: lexical, grammatical, semantic and positional.

Once we have identified entity classes, the next step is to fill out grid entries with relevant syntactic information. We employ a robust statistical parser (Collins, 1997) to determine the constituent structure for each sentence, from which subjects (**S**), objects (**O**), and relations other than subject or object (**X**) are identified. Passive verbs are recognized using a small set of patterns, and the underlying deep grammatical role for arguments involved in the passive construction is entered in the grid (see the grid cell **O** for *Microsoft*, Sentence 2, Table 2).

When a noun is attested more than once with a different grammatical role in the same sentence, we default to the role with the highest grammatical ranking: subjects are ranked higher than objects, which in turn are ranked higher than the rest. For example, the entity *Microsoft* is mentioned twice in Sentence 1 with the grammatical roles **X** (for *Microsoft Corp.*) and **S** (for *the company*), but is represented only by **S** in the grid (see Tables 1 and 2).

Coherence Assessment We introduce a method for coherence assessment that is based on grid representation. A fundamental assumption underlying our approach is that the distribution of entities in coherent texts exhibits certain regularities reflected in grid topology. Some of these regularities are formalized in Centering Theory as constraints on transitions of local focus in adjacent sentences. Grids of coherent texts are likely to have some dense columns (i.e., columns with just a few gaps such as *Microsoft* in Table 1) and many sparse columns which will consist mostly of gaps (see *markets*, *earnings* in Table 1). One would further expect that entities corresponding to dense columns are more often subjects or objects. These characteristics will be less pronounced in low-coherence texts.

Inspired by Centering Theory, our analysis revolves around patterns of local entity transitions. A *local entity transition* is a sequence $\{\mathbf{S},\mathbf{O},\mathbf{X},\mathbf{–}\}^n$ that represents entity occurrences and their syntactic roles in *n* adjacent sentences. Local transitions can be easily obtained from a grid as continuous subsequences of each column. Each transition will have a certain probability in a given grid. For instance, the probability of the transition **[S –]** in the grid from Table 1 is 0.08 (computed as a ratio of its frequency (i.e., six) divided by the total number of transitions of length two (i.e., 75)). Each text can thus be viewed as a distribution defined over transition types. We believe that considering all entity transitions may uncover new patterns relevant for coherence assessment.

We further refine our analysis by taking into account the salience of discourse entities. Centering and other discourse theories conjecture that the way an entity is introduced and mentioned depends on its global role in a given discourse. Therefore, we discriminate between transitions of salient entities and the rest, collecting statistics for each group separately. We identify salient entities based on their

	S S	S O	S X	S –	O S	O O	O X	O –	X S	X O	X X	X –	– S	– O	– X	– –
d_1	0	0	0	.03	0	0	0	.02	.07	0	0	.12	.02	.02	.05	.25
d_2	0	0	0	.02	0	.07	0	.02	0	0	.06	.04	0	0	0	.36
d_3	.02	0	0	.03	0	0	0	.06	0	0	0	.05	.03	.07	.07	.29

Table 3: Example of a feature-vector document representation using all transitions of length two given syntactic categories: **S**, **O**, **X**, and **–**.

frequency,[1] following the widely accepted view that the occurrence frequency of an entity correlates with its discourse prominence (Morris and Hirst, 1991; Grosz et al., 1995).

Ranking We view coherence assessment as a ranking learning problem. The ranker takes as input a set of alternative renderings of the same document and ranks them based on their degree of local coherence. Examples of such renderings include a set of different sentence orderings of the same text and a set of summaries produced by different systems for the same document. Ranking is more suitable than classification for our purposes since in text generation, a system needs a scoring function to compare among alternative renderings. Furthermore, it is clear that coherence assessment is not a categorical decision but a graded one: there is often no single coherent rendering of a given text but many different possibilities that can be partially ordered.

As explained previously, coherence constraints are modeled in the grid representation implicitly by entity transition sequences. To employ a machine learning algorithm to our problem, we encode transition sequences explicitly using a standard feature vector notation. Each grid rendering j of a document d_i is represented by a feature vector $\Phi(x_{ij}) = (p_1(x_{ij}), p_2(x_{ij}), \ldots, p_m(x_{ij}))$, where m is the number of all predefined entity transitions, and $p_t(x_{ij})$ the probability of transition t in grid x_{ij}. Note that considerable latitude is available when specifying the transition types to be included in a feature vector. These can be all transitions of a given length (e.g., two or three) or the most frequent transitions within a document collection. An example of a feature space with transitions of length two is illustrated in Table 3.

The training set consists of ordered pairs of renderings (x_{ij}, x_{ik}), where x_{ij} and x_{ik} are renderings of the same document d_i, and x_{ij} exhibits a higher degree of coherence than x_{ik}. Without loss of generality, we assume $j > k$. The goal of the training procedure is to find a parameter vector $\vec{w}$ that yields a "ranking score" function $\vec{w} \cdot \Phi(x_{ij})$, which minimizes the number of violations of pairwise rankings provided in the training set. Thus, the ideal $\vec{w}$ would satisfy the condition $\vec{w} \cdot (\Phi(x_{ij}) - \Phi(x_{ik})) > 0 \ \forall j, i, k$ such that $j > k$. The problem is typically treated as a Support Vector Machine constraint optimization problem, and can be solved using the search technique described in Joachims (2002a). This approach has been shown to be highly effective in various tasks ranging from collaborative filtering (Joachims, 2002a) to parsing (Toutanova et al., 2004).

In our ranking experiments, we use Joachims' (2002a) SVM^{light} package for training and testing with all parameters set to their default values.

4 Evaluation Set-Up

In this section we describe two evaluation tasks that assess the merits of the coherence modeling framework introduced above. We also give details regarding our data collection, and parameter estimation. Finally, we introduce the baseline method used for comparison with our approach.

4.1 Text Ordering

Text structuring algorithms (Lapata, 2003; Barzilay and Lee, 2004; Karamanis et al., 2004) are commonly evaluated by their performance at information-ordering. The task concerns determining a sequence in which to present a pre-selected set of information-bearing items; this is an essential step in concept-to-text generation, multi-document summarization, and other text-synthesis problems. Since local coherence is a key property of any well-formed text, our model can be used to rank alternative sentence orderings. We do not assume that local coherence is sufficient to uniquely determine the best ordering — other constraints clearly play a role here. However, we expect that the accuracy of a coherence model is reflected in its performance in the ordering task.

Data To acquire a large collection for training and testing, we create synthetic data, wherein the candidate set consists of a source document and permutations of its sentences. This framework for data acquisition is widely used in evaluation of ordering algorithms as it enables large scale automatic evalu-

[1]The frequency threshold is empirically determined on the development set. See Section 5 for further discussion.

ation. The underlying assumption is that the original sentence order in the source document must be coherent, and so we should prefer models that rank it higher than other permutations. Since we do not know the relative quality of different permutations, our corpus includes only pairwise rankings that comprise the original document and one of its permutations. Given k original documents, each with n randomly generated permutations, we obtain $k \cdot n$ (trivially) annotated pairwise rankings for training and testing.

Using the technique described above, we collected data in two different genres: newspaper articles and accident reports written by government officials. The first collection consists of Associated Press articles from the North American News Corpus on the topic of natural disasters. The second includes narratives from the National Transportation Safety Board's database[2]. Both sets have documents of comparable length – the average number of sentences is 10.4 and 11.5, respectively. For each set, we used 100 source articles with 20 randomly generated permutations for training. The same number of pairwise rankings (i.e., 2000) was used for testing. We held out 10 documents (i.e., 200 pairwise rankings) from the training data for development purposes.

4.2 Summary Evaluation

We further test the ability of our method to assess coherence by comparing model induced rankings against rankings elicited by human judges. Admittedly, the information ordering task only partially approximates degrees of coherence violation using different sentence permutations of a source document. A stricter evaluation exercise concerns the assessment of texts with naturally occurring coherence violations as perceived by human readers. A representative example of such texts are automatically generated summaries which often contain sentences taken out of context and thus display problems with respect to local coherence (e.g., dangling anaphors, thematically unrelated sentences). A model that exhibits high agreement with human judges not only accurately captures the coherence properties of the summaries in question, but ultimately holds promise for the automatic evaluation of machine-generated texts. Existing automatic evaluation measures such as BLEU (Papineni et al., 2002) and ROUGE (Lin and Hovy, 2003), are not designed for the coherence assessment task, since they focus on content similarity between system output and reference texts.

Data Our evaluation was based on materials from the Document Understanding Conference (DUC, 2003), which include multi-document summaries produced by human writers and by automatic summarization systems. In order to learn a ranking, we require a set of summaries, each of which have been rated in terms of coherence. We therefore elicited judgments from human subjects.[3] We randomly selected 16 input document clusters and five systems that had produced summaries for these sets, along with summaries composed by several humans. To ensure that we do not tune a model to a particular system, we used the output summaries of distinct systems for training and testing. Our set of training materials contained $4 \cdot 16$ summaries (average length 4.8), yielding $\binom{4}{2} \cdot 16 = 96$ pairwise rankings. In a similar fashion, we obtained 32 pairwise rankings for the test set. Six documents from the training data were used as a development set.

Coherence ratings were obtained during an elicitation study by 177 unpaid volunteers, all native speakers of English. The study was conducted remotely over the Internet. Participants first saw a set of instructions that explained the task, and defined the notion of coherence using multiple examples. The summaries were randomized in lists following a Latin square design ensuring that no two summaries in a given list were generated from the same document cluster. Participants were asked to use a seven point scale to rate how coherent the summaries were without having seen the source texts. The ratings (approximately 23 per summary) given by our subjects were averaged to provide a rating between 1 and 7 for each summary.

The reliability of the collected judgments is crucial for our analysis; we therefore performed several tests to validate the quality of the annotations. First, we measured how well humans agree in their coherence assessment. We employed leave-one-out resampling[4] (Weiss and Kulikowski, 1991), by correlating the data obtained from each participant with the mean coherence ratings obtained from all other participants. The inter-subject agree-

[2]The collections are available from `http://www.csail.mit.edu/regina/coherence/`.

[3]The ratings are available from `http://homepages.inf.ed.ac.uk/mlap/coherence/`.

[4]We cannot apply the commonly used Kappa statistic for measuring agreement since it is appropriate for nominal scales, whereas our summaries are rated on an ordinal scale.

ment was $r = .768$. Second, we examined the effect of different types of summaries (human- vs. machine-generated.) An ANOVA revealed a reliable effect of summary type: $F(1;15) = 20.38$, $p < 0.01$ indicating that human summaries are perceived as significantly more coherent than system-generated ones. Finally, the judgments of our participants exhibit a significant correlation with DUC evaluations ($r = .41$, $p < 0.01$).

4.3 Parameter Estimation

Our model has two free parameters: the frequency threshold used to identify salient entities and the length of the transition sequence. These parameters were tuned separately for each data set on the corresponding held-out development set. For our ordering and summarization experiments, optimal salience-based models were obtained for entities with frequency ≥ 2. The optimal transition length was ≤ 3 for ordering and ≤ 2 for summarization.

4.4 Baseline

We compare our algorithm against the coherence model proposed by Foltz et al. (1998) which measures coherence as a function of semantic relatedness between adjacent sentences. Semantic relatedness is computed automatically using Latent Semantic Analysis (LSA, Landauer and Dumais 1997) from raw text without employing syntactic or other annotations. This model is a good point of comparison for several reasons: (a) it is fully automatic, (b) it is a not a straw-man baseline; it correlates reliably with human judgments and has been used to analyze discourse structure, and (c) it models an aspect of coherence which is orthogonal to ours (their model is lexicalized).

Following Foltz et al. (1998) we constructed vector-based representations for individual words from a lemmatized version of the North American News Text Corpus[5] (350 million words) using a term-document matrix. We used singular value decomposition to reduce the semantic space to 100 dimensions obtaining thus a space similar to LSA. We represented the meaning of a sentence as a vector by taking the mean of the vectors of its words. The similarity between two sentences was determined by measuring the cosine of their means. An overall text coherence measure was obtained by averaging the cosines for all pairs of adjacent sentences.

[5]Our selection of this corpus was motivated by its similarity to the DUC corpus which primarily consists of news stories.

In sum, each text was represented by a single feature, its sentence-to-sentence semantic similarity. During training, the ranker learns an appropriate threshold value for this feature.

4.5 Evaluation Metric

Model performance was assessed in the same way for information ordering and summary evaluation. Given a set of pairwise rankings, we measure accuracy as the ratio of correct predictions made by the model over the size of the test set. In this setup, random prediction results in an accuracy of 50%.

5 Results

The evaluation of our coherence model was driven by two questions: (1) How does the proposed model compare to existing methods for coherence assessment that make use of distinct representations? (2) What is the contribution of linguistic knowledge to the model's performance? Table 4 summarizes the accuracy of various configurations of our model for the ordering and coherence assessment tasks.

We first compared a linguistically rich grid model that incorporates coreference resolution, expressive syntactic information, and a salience-based feature space (Coreference+Syntax+Salience) against the LSA baseline (LSA). As can be seen in Table 4, the grid model outperforms the baseline in both ordering and summary evaluation tasks, by a wide margin. We conjecture that this difference in performance stems from the ability of our model to discriminate between various patterns of local sentence transitions. In contrast, the baseline model only measures the degree of overlap across successive sentences, without taking into account the properties of the entities that contribute to the overlap. Not surprisingly, the difference between the two methods is more pronounced for the second task — summary evaluation. Manual inspection of our summary corpus revealed that low-quality summaries often contain repetitive information. In such cases, simply knowing about high cross-sentential overlap is not sufficient to distinguish a repetitive summary from a well-formed one.

In order to investigate the contribution of linguistic knowledge on model performance we compared the full model introduced above against models using more impoverished representations. We focused on three sources of linguistic knowledge — syntax, coreference resolution, and salience — which play

Model	Ordering (Set1)	Ordering (Set2)	Summarization
Coreference+Syntax+Salience	**87.3**	**90.4**	**68.8**
Coreference+Salience	86.9	88.3	62.5
Syntax+Salience	83.4	89.7	81.3
Coreference+Syntax	76.5	88.8	75.0
LSA	72.1	72.1	25.0

Table 4: Ranking accuracy measured as the fraction of correct pairwise rankings in the test set.

a prominent role in Centering analyses of discourse coherence. An additional motivation for our study is exploration of the trade-off between robustness and richness of linguistic annotations. NLP tools are typically trained on human-authored texts, and may deteriorate in performance when applied to automatically generated texts with coherence violations.

Syntax To evaluate the effect of syntactic knowledge, we eliminated the identification of grammatical relations from our grid computation and recorded solely whether an entity is present or absent in a sentence. This leaves only the coreference and salience information in the model, and the results are shown in Table 4 under (Coreference+Salience). The omission of syntactic information causes a uniform drop in performance on both tasks, which confirms its importance for coherence analysis.

Coreference To measure the effect of fully-fledged coreference resolution, we constructed entity classes simply by clustering nouns on the basis of their identity. In other words, each noun in a text corresponds to a different entity in a grid, and two nouns are considered coreferent only if they are identical. The performance of the model (Syntax+Salience) is shown in the third row of Table 4.

While coreference resolution improved model performance in ordering, it caused a decrease in accuracy in summary evaluation. This drop in performance can be attributed to two factors related to the nature of our corpus — machine-generated texts. First, an automatic coreference resolution tool expectedly decreases in accuracy because it was trained on well-formed human-authored texts. Second, automatic summarization systems do not use anaphoric expressions as often as humans do. Therefore, a simple entity clustering method is more suitable for automatic summaries.

Salience Finally, we evaluate the contribution of salience information by comparing our original model (Coreference+Syntax+Salience) which accounts separately for patterns of salient and non-salient entities against a model that does not attempt to discriminate between them (Coreference+Syntax). Our results on the ordering task indicate that models that take salience information into account consistently outperform models that do not. The effect of salience is less pronounced for the summarization task when it is combined with coreference information (Coreference + Salience). This is expected, since accurate identification of coreferring entities is prerequisite to deriving accurate salience models. However, as explained above, our automatic coreference tool introduces substantial noise in our representation. Once this noise is removed (see Syntax+Salience), the salience model has a clear advantage over the other models.

6 Discussion and Conclusions

In this paper we proposed a novel framework for representing and measuring text coherence. Central to this framework is the *entity grid* representation of discourse which we argue captures important patterns of sentence transitions. We re-conceptualize coherence assessment as a ranking task and show that our entity-based representation is well suited for learning an appropriate ranking function; we achieve good performance on text ordering and summary coherence evaluation.

On the linguistic side, our results yield empirical support to some of Centering Theory's main claims. We show that coherent texts are characterized by transitions with particular properties which do not hold for all discourses. Our work, however, not only validates these findings, but also quantitatively measures the predictive power of various linguistic features for the task of coherence assessment.

An important future direction lies in augmenting our entity-based model with lexico-semantic knowledge. One way to achieve this goal is to cluster entities based on their semantic relatedness, thereby creating a grid representation over lexical chains (Morris and Hirst, 1991). An entirely different approach is to develop fully lexicalized models, akin to traditional language models. Cache language models (Kuhn and Mori, 1990) seem particularly promising in this context.

In the discourse literature, entity-based theories are primarily applied at the level of local coherence, while relational models, such as Rhetorical Structure Theory (Mann and Thomson, 1988; Marcu, 2000), are used to model the global structure of discourse. We plan to investigate how to combine the two for improved prediction on both local and global levels, with the ultimate goal of handling longer texts.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (Barzilay; CAREER grant IIS-0448168 and grant IIS-0415865) and EPSRC (Lapata; grant GR/T04540/01). We are grateful to Claire Cardie and Vincent Ng for providing us the results of their system on our data. Thanks to Eli Barzilay, Eugene Charniak, Michael Elhadad, Noemie Elhadad, Frank Keller, Alex Lascarides, Igor Malioutov, Smaranda Muresan, Martin Rinard, Kevin Simler, Caroline Sporleder, Chao Wang, Bonnie Webber and three anonymous reviewers for helpful comments and suggestions. Any opinions, findings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the National Science Foundation or EPSRC.

References

N. Asher, A. Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

R. Barzilay, L. Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of HLT-NAACL*, 113–120.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the ACL/EACL*, 16–23.

P. W. Foltz, W. Kintsch, T. K. Landauer. 1998. Textual coherence using latent semantic analysis. *Discourse Processes*, 25(2&3):285–307.

B. Grosz, A. K. Joshi, S. Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

L. Hasler. 2004. An investigation into the use of centering transitions for summarisation. In *Proceedings of the 7th Annual CLUK Research Colloquium*, 100–107, University of Birmingham.

T. Joachims. 2002a. Optimizing search engines using clickthrough data. In *Proceesings of KDD*, 133–142.

N. Karamanis, M. Poesio, C. Mellish, J. Oberlander. 2004. Evaluating centering-based metrics of coherence for text structuring using a reliably annotated corpus. In *Proceedings of the ACL*, 391–398.

R. Kibble, R. Power. 2004. Optimising referential coherence in text generation. *Computational Linguistics*, 30(4):401–416.

R. Kuhn, R. D. Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transactions on PAMI*, 12(6):570–583.

T. K. Landauer, S. T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

M. Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the ACL*, 545–552.

C.-Y. Lin, E. H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT-NAACL*, 71–78.

W. C. Mann, S. A. Thomson. 1988. Rhetorical structure theory. *Text*, 8(3):243–281.

D. Marcu. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press.

E. Miltsakaki, K. Kukich. 2000. The role of centering theory's rough-shift in the teaching and evaluation of writing skills. In *Proceedings of the ACL*, 408–415.

J. Morris, G. Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 1(17):21–43.

V. Ng, C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the ACL*, 104–111.

K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the ACL*, 311–318.

M. Poesio, R. Stevenson, B. D. Eugenio, J. Hitzeman. 2004. Centering: a parametric theory and its instantiations. *Computational Linguistics*, 30(3):309–363.

E. Reiter, R. Dale. 2000. *Building Natural-Language Generation Systems*. Cambridge University Press.

D. Scott, C. S. de Souza. 1990. Getting the message across in RST-based text generation. In R. Dale, C. Mellish, M. Zock, eds., *Current Research in Natural Language Generation*, 47–73. Academic Press.

M. Strube, U. Hahn. 1999. Functional centering – grounding referential coherence in information structure. *Computational Linguistics*, 25(3):309–344.

K. Toutanova, P. Markova, C. D. Manning. 2004. The leaf projection path view of parse trees: Exploring string kernels for HPSG parse selection. In *Proceedings of the EMNLP*, 166–173.

M. Walker, A. Joshi, E. Prince, eds. 1998. *Centering Theory in Discourse*. Clarendon Press.

S. M. Weiss, C. A. Kulikowski. 1991. *Computer Systems that Learn: Classification and Prediction Methods from, Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann.

Modelling the substitutability of discourse connectives

Ben Hutchinson
School of Informatics
University of Edinburgh
B.Hutchinson@sms.ed.ac.uk

Abstract

Processing discourse connectives is important for tasks such as discourse parsing and generation. For these tasks, it is useful to know which connectives can signal the same coherence relations. This paper presents experiments into modelling the substitutability of discourse connectives. It shows that substitutability effects distributional similarity. A novel variance-based function for comparing probability distributions is found to assist in predicting substitutability.

1 Introduction

Discourse coherence relations contribute to the meaning of texts, by specifying the relationships between semantic objects such as events and propositions. They also assist in the interpretation of anaphora, verb phrase ellipsis and lexical ambiguities (Hobbs, 1985; Kehler, 2002; Asher and Lascarides, 2003). Coherence relations can be implicit, or they can be signalled explicitly through the use of discourse connectives, e.g. *because*, *even though*.

For a machine to interpret a text, it is important that it recognises coherence relations, and so as explicit markers discourse connectives are of great assistance (Marcu, 2000). When discourse connectives are not present, the task is more difficult. For such cases, unsupervised approaches have been developed for predicting relations, by using sentences containing discourse connectives as training data (Marcu and Echihabi, 2002; Lapata and Lascarides, 2004). However the nature of the relationship between the coherence relations signalled by discourse connectives and their empirical distributions has to date been poorly understood. In particular, one might wonder whether connectives with similar meanings also have similar distributions.

Concerning natural language generation, texts are easier for humans to understand if they are coherently structured. Addressing this, a body of research has considered the problems of generating appropriate discourse connectives (for example (Moser and Moore, 1995; Grote and Stede, 1998)). One such problem involves choosing which connective to generate, as the mapping between connectives and relations is not one-to-one, but rather many-to-many. Siddharthan (2003) considers the task of paraphrasing a text while preserving its rhetorical relations. Clauses conjoined by *but*, *or* and *when* are separated to form distinct orthographic sentences, and these conjunctions are replaced by the discourse adverbials *however*, *otherwise* and *then*, respectively.

The idea underlying Siddharthan's work is that one connective can be substituted for another while preserving the meaning of a text. Knott (1996) studies the substitutability of discourse connectives, and proposes that substitutability can motivate theories of discourse coherence. Knott uses an empirical methodology to determine the substitutability of pairs of connectives. However this methodology is manually intensive, and Knott derives relationships for only about 18% of pairs of connectives. It would thus be useful if substitutability could be predicted automatically.

This paper proposes that substitutability can be predicted through statistical analysis of the contexts in which connectives appear. Similar methods have been developed for predicting the similarity of nouns and verbs on the basis of their distributional similarity, and many distributional similarity functions have been proposed for these tasks (Lee, 1999). However substitutability is a more complex notion than similarity, and we propose a novel variance-based function for assisting in this task.

This paper constitutes a first step towards predicting substitutability of cnonectives automatically. We demonstrate that the substitutability of connectives has significant effects on both distributional similarity and the new variance-based function. We then attempt to predict substitutability of connectives using a simplified task that factors out the prior likelihood of being substitutable.

2 Relationships between connectives

Two types of relationships between connectives are of interest: similarity and substitutability.

2.1 Similarity

The concept of lexical similarity occupies an important role in psychology, artificial intelligence, and computational linguistics. For example, in psychology, Miller and Charles (1991) report that psychologists 'have largely abandoned "synonymy" in favour of "semantic similarity".' In addition, work in automatic lexical acquisition is based on the proposition that distributional similarity correlates with semantic similarity (Grefenstette, 1994; Curran and Moens, 2002; Weeds and Weir, 2003).

Several studies have found subjects' judgements of semantic similarity to be robust. For example, Miller and Charles (1991) elicit similarity judgements for 30 pairs of nouns such as *cord–smile*, and found a high correlation with judgements of the same data obtained over 25 years previously (Rubenstein and Goodenough, 1965). Resnik (1999) repeated the experiment, and calculated an inter-rater agreement of 0.90. Resnik and Diab (2000) also performed a similar experiment with pairs of verbs (e.g. *bathe–kneel*). The level of inter-rater agreement was again significant ($r = 0.76$).

1. Take an instance of a discourse connective in a corpus. Imagine you are the writer that produced this text, but that you need to choose an alternative connective.
2. Remove the connective from the text, and insert another connective in its place.
3. If the new connective achieves the same discourse goals as the original one, it is considered **substitutable** in this context.

Figure 1: Knott's Test for Substitutability

Given two words, it has been suggested that if words have the similar meanings, then they can be expected to have similar contextual distributions. The studies listed above have also found evidence that similarity ratings correlate positively with the distributional similarity of the lexical items.

2.2 Substitutability

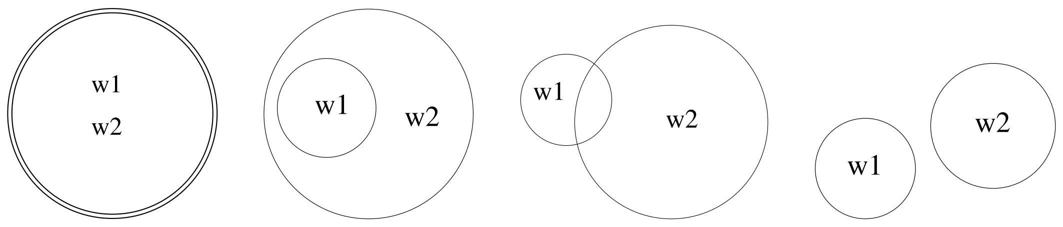
The notion of substitutability has played an important role in theories of lexical relations. A definition of synonymy attributed to Leibniz states that two words are synonyms if one word can be used in place of the other without affecting truth conditions.

Unlike similarity, the substitutability of discourse connectives has been previously studied. Halliday and Hasan (1976) note that in certain contexts *otherwise* can be paraphrased by *if not*, as in

(1) It's the way I like to go to work.
One person and one line of enquiry at a time.
Otherwise/if not, there's a muddle.

They also suggest some other extended paraphrases of *otherwise*, such as *under other circumstances*.

Knott (1996) systematises the study of the substitutability of discourse connectives. His first step is to propose a Test for Substitutability for connectives, which is summarised in Figure 1. An application of the Test is illustrated by (2). Here *seeing as* was the connective originally used by the writer, however *because* can be used instead.



(a) w_1 and w_2 are SYNONYMS

(b) w_1 is a HYPONYM of w_2

(c) w_1 and w_2 are CONTINGENTLY SUBSTITUTABLE

(d) w_1 and w_2 are EXCLUSIVE

Figure 2: Venn diagrams representing relationships between distributions

(2) **Seeing as/because** we've got nothing but circumstantial evidence, it's going to be difficult to get a conviction. (Knott, p. 177)

However the ability to substitute is sensitive to the context. In other contexts, for example (3), the substitution of *because* for *seeing as* is not valid.

(3) It's a fairly good piece of work, **seeing as/#because** you have been under a lot of pressure recently. (Knott, p. 177)

Similarly, there are contexts in which *because* can be used, but *seeing as* cannot be substituted for it:

(4) That proposal is useful, **because/#seeing as** it gives us a fallback position if the negotiations collapse. (Knott, p. 177)

Knott's next step is to generalise over all contexts a connective appears in, and to define four substitutability relationships that can hold between a pair of connectives w_1 and w_2. These relationships are illustrated graphically through the use of Venn diagrams in Figure 2, and defined below.

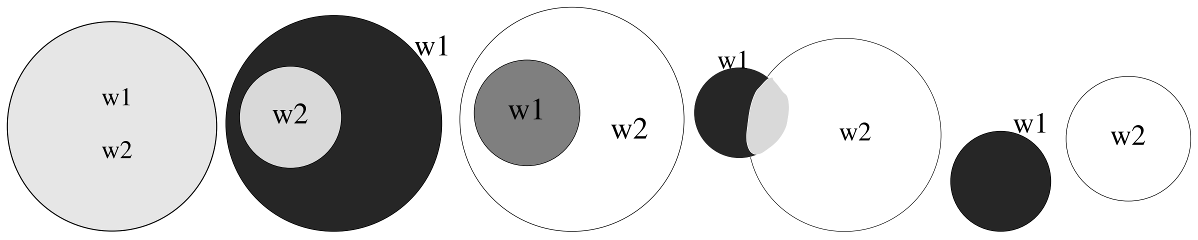
- w_1 is a SYNONYM of w_2 if w_1 can always be substituted for w_2, and vice versa.
- w_1 and w_2 are EXCLUSIVE if neither can ever be substituted for the other.
- w_1 is a HYPONYM of w_2 if w_2 can always be substituted for w_1, but not vice versa.
- w_1 and w_2 are CONTINGENTLY SUBSTITUTABLE if each can sometimes, but not always, be substituted for the other.

Given examples (2)–(4) we can conclude that *because* and *seeing as* are CONTINGENTLY SUBSTITUTABLE (henceforth "CONT. SUBS."). However this is the only relationship that can be established using a finite number of linguistic examples. The other relationships all involve generalisations over all contexts, and so rely to some degree on the judgement of the analyst. Examples of each relationship given by Knott (1996) include: *given that* and *seeing as* are SYNONYMS, *on the grounds that* is a HYPONYM of *because*, and *because* and *now that* are EXCLUSIVE.

Although substitutability is inherently a more complex notion than similarity, distributional similarity is expected to be of some use in predicting substitutability relationships. For example, if two discourse connectives are SYNONYMS then we would expect them to have similar distributions. On the other hand, if two connectives are EXCLUSIVE, then we would expect them to have dissimilar distributions. However if the relationship between two connectives is HYPONYMY or CONT. SUBS. then we expect to have partial overlap between their distributions (consider Figure 2), and so distributional similarity might not distinguish these relationships.

The Kullback-Leibler (KL) divergence function is a distributional similarity function that is of particular relevance here since it can be described informally in terms of substitutability. Given co-occurrence distributions p and q, its mathematical definition can be written as:

$$D(p||q) = \sum_x p(x)(\log \frac{1}{q(x)} - \log \frac{1}{p(x)}) \qquad (5)$$



(a) w_1 and w_2 are SYNONYMS (b) w_2 is a HYPONYM of w_1 (c) w_1 is a HYPONYM of w_2 (d) w_1 and w_2 are CONT. SUBS. (e) w_1 and w_2 are EXCLUSIVE

Figure 3: Surprise in substituting w_2 for w_1 (darker shading indicates higher surprise)

The value $\log \frac{1}{p(x)}$ has an informal interpretation as a measure of how surprised an observer would be to see event x, given prior likelihood expectations defined by p. Thus, if p and q are the distributions of words w_1 and w_2 then

$$D(p||q) = E_p(\text{surprise in seeing } w_2 - \text{surprise in seeing } w_1) \quad (6)$$

where E_p is the expectation function over the distribution of w_1 (i.e. p). That is, KL divergence measures how much more surprised we would be, on average, to see word w_2 rather than w_1, where the averaging is weighted by the distribution of w_1.

3 A variance-based function for distributional analysis

A distributional similarity function provides only a one-dimensional comparison of two distributions, namely how similar they are. However we can obtain an additional perspective by using a variance-based function. We now introduce a new function V by taking the variance of the surprise in seeing w_2, over the contexts in which w_1 appears:

$$\begin{aligned} V(p,q) &= Var(\text{surprise in seeing } w_2) \\ &= E_p((E_p(\log \frac{1}{q(x)}) - \log \frac{1}{q(x)})^2) \quad (7) \end{aligned}$$

Note that like KL divergence, $V(p,q)$ is asymmetric.

We now consider how the substitutability of connectives affects our expectations of the value of V. If two connectives are SYNONYMS then each can always be used in place of other. Thus we would always expect a low level of surprise in seeing one connective in place of the other, and this low level of surprise is indicated via light shading in Figure 3a. It follows that the variance in surprise is low. On the other hand, if two connectives are EXCLUSIVE then there would always be a high degree of surprise in seeing one in place of the other. This is indicated using dark shading in Figure 3e. Only one set is shaded because we need only consider the contexts in which w_1 is appropriate. In this case, the variance in surprise is again low. The situation is more interesting when we consider two connectives that are CONT. SUBS.. In this case substitutability (and hence surprise) is dependent on the context. This is illustrated using light and dark shading in Figure 3d. As a result, the variance in surprise is high. Finally, with HYPONYMY, the variance in surprise depends on whether the original connective was the HYPONYM or the HYPERNYM.

Relationship	Function			
of w_1 to w_2	$D(p\|\|q)$	$D(q\|\|p)$	$V(p,q)$	$V(q,p)$
SYNONYM	Low	Low	Low	Low
HYPONYM	Low	Medium	Low	High
CONT. SUBS.	Medium	Medium	High	High
EXCLUSIVE	High	High	Low	Low

Table 1: Expectations for distributional functions

Table 1 summarises our expectations of the values of KL divergence and V, for the various substitutability relationships. (KL divergence, unlike most similarity functions, is sensitive to the order of arguments related by hyponymy (Lee, 1999).) The

Something happened **and** *something else happened.*
Something happened **or** *something else happened.*
◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5

Figure 4: Example experimental item

	Mean	HYP	CONT. SUBS.	EXCL
SYNONYM	3.97	*	*	*
HYPONYM	3.43		*	*
CONT. SUBS.	1.79			*
EXCLUSIVE	1.08			

Table 2: Similarity by substitutability relationship

experiments described below test these expectations using empirical data.

4 Experiments

We now describe our empirical experiments which investigate the connections between a) subjects' ratings of the similarity of discourse connectives, b) the substitutability of discourse connectives, and c) KL divergence and the new function V applied to the distributions of connectives. Our motivation is to explore how distributional properties of words might be used to predict substitutability. The experiments are restricted to connectives which relate clauses within a sentence. These include coordinating conjunctions (e.g. *but*) and a range of subordinators including conjunctions (e.g. *because*) as well as phrases introducing adverbial clauses (e.g. *now that, given that, for the reason that*). Adverbial discourse connectives are therefore not considered.

4.1 Experiment 1: Subject ratings of similarity

This experiment tests the hypotheses that 1) subjects agree on the degree of similarity between pairs of discourse connectives, and 2) similarity ratings correlate with the degree of substitutability.

4.1.1 Methodology

We randomly selected 48 pairs of discourse connectives such that there were 12 pairs standing in each of the four substitutability relationships.To do this, we used substitutability judgements made by Knott (1996), supplemented with some judgements of our own. Each experimental item consisted of the two discourse connectives along with dummy clauses, as illustrated in Figure 4. The format of the experimental items was designed to indicate how a phrase could be used as a discourse connective (e.g. it may not be obvious to a subject that the phrase *the moment* is a discourse connective), but without providing complete semantics for the clauses, which might bias the subjects' ratings. Forty native speakers of English participated in the experiment, which was conducted remotely via the internet.

4.1.2 Results

Leave-one-out resampling was used to compare each subject's ratings are with the means of their peers' (Weiss and Kulikowski, 1991). The average inter-subject correlation was 0.75 (Min = 0.49, Max = 0.86, StdDev = 0.09), which is comparable to previous results on verb similarity ratings (Resnik and Diab, 2000). The effect of substitutability on similarity ratings can be seen in Table 2. Post-hoc Tukey tests revealed all differences between means in Table 2 to be significant.

The results demonstrate that subjects' ratings of connective similarity show significant agreement and are robust enough for effects of substitutability to be found.

4.2 Experiment 2: Modelling similarity

This experiment compares subjects' ratings of similarity with lexical co-occurrence data. It hypothesises that similarity ratings correlate with distributional similarity, but that neither correlates with the new variance in surprise function.

4.2.1 Methodology

Sentences containing discourse connectives were gathered from the British National Corpus and the world wide web, with discourse connectives identified on the basis of their syntactic contexts (for details, see Hutchinson (2004b)). The mean number of sentences per connective was about $32,000$, although about 12% of these are estimated to be errors. From these sentences, lexical co-occurrence data were collected. Only co-occurrences with dis-

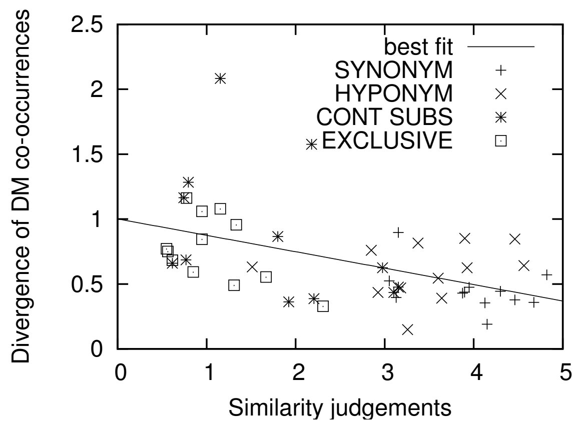


Figure 5: Similarity versus distributional divergence

course adverbials and other structural discourse connectives were stored, as these had previously been found to be useful for predicting semantic features of connectives (Hutchinson, 2004a).

4.2.2 Results

A skewed variant of the Kullback-Leibler divergence function was used to compare co-occurrence distributions (Lee, 1999, with $\alpha = 0.95$). Spearman's correlation coefficient for ranked data showed a significant correlation ($r = -0.51$, $p < 0.001$). (The correlation is negative because KL divergence is lower when distributions are more similar.) The strength of this correlation is comparable with similar results achieved for verbs (Resnik and Diab, 2000), but not as great as has been observed for nouns (McDonald, 2000). Figure 5 plots the mean similarity judgements against the distributional divergence obtained using discourse markers, and also indicates the substitutability relationship for each item. (Two outliers can be observed in the upper left corner; these were excluded from the calculations.)

The "variance in surprise" function introduced in the previous section was applied to the same co-occurrence data.[1] These variances were compared to distributional divergence and the subjects' similarity ratings, but in both cases Spearman's correlation coefficient was not significant.

In combination with the previous experiment, these results demonstrate a three way correspondence between the human ratings of the similarity of a pair of connectives, their substitutability relationship, and their distributional similarity. Hutchinson (2005) presents further experiments on modelling connective similarity, and discusses their implications. This experiment also provides empirical evidence that the new variance in surprise function is not a measure of similarity.

4.3 Experiment 3: Predicting substitutability

The previous experiments provide hope that substitutability of connectives might be predicted on the basis of their empirical distributions. However one complicating factor is that EXCLUSIVE is by far the most likely relationship, holding between about 70% of pairs. Preliminary experiments showed that the empirical evidence for other relationships was not strong enough to overcome this prior bias. We therefore attempted two pseudodisambiguation tasks which eliminated the effects of prior likelihoods. The first task involved distinguishing between the relationships whose connectives subjects rated as most similar, namely SYNONYMY and HYPONYMY. Triples of connectives $\langle p, q, q' \rangle$ were collected such that SYNONYM(p, q) and either HYPONYM(p, q') or HYPONYM(q', p) (we were not attempting to predict the order of HYPONYMY). The task was then to decide automatically which of q and q' is the SYNONYM of p.

The second task was identical in nature to the first, however here the relationship between p and q was either SYNONYMY or HYPONYMY, while p and q' were either CONT. SUBS. or EXCLUSIVE. These two sets of relationships are those corresponding to high and low similarity, respectively. In combination, the two tasks are equivalent to predicting SYNONYMY or HYPONYMY from the set of all four relationships, by first distinguishing the high similarity relationships from the other two, and then making a finer-grained distinction between the two.

4.3.1 Methodology

Substitutability relationships between 49 structural discourse connectives were extracted from Knott's (1996) classification. In order to obtain more evaluation data, we used Knott's methodology to obtain relationships between an additional 32 connec-

[1]In practice, the skewed variant $V(p, 0.95q + 0.05p)$ was used, in order to avoid problems arising when $q(x) = 0$.

	$max(D_1, D_2)$	$max(V_1, V_2)$	$(V_1 - V_2)^2$
SYN	0.627	4.44	3.29
HYP	0.720	5.16	8.02
CONT	1.057	4.85	7.81
EXCL	1.069	4.79	7.27

Table 3: Distributional analysis by substitutability

Input to Gaussian Model	SYN VS HYP	SYN/HYP VS EX/CONT
$max(D_1, D_2)$	50.0%	76.1%
$max(V_1, V_2)$	84.8%	60.6%

Table 4: Accuracy on pseudodisambiguation task

tives. This resulted in 46 triples $\langle p, q, q' \rangle$ for the first task, and 10,912 triples for the second task.

The co-occurrence data from the previous section were re-used. These were used to calculate $D(p||q)$ and $V(p, q)$. Both of these are asymmetric, so for our purposes we took the maximum of applying their arguments in both orders. Recall from Table 1 that when two connectives are in a HYPONYMY relation we expect V to be sensitive to the order in which the connectives are given as arguments. To test this, we also calculated $(V(p, q) - V(q, p))^2$, i.e. the square of the difference of applying the arguments to V in both orders. The average values are summarised in Table 3, with D_1 and D_2 (and V_1 and V_2) denoting different orderings of the arguments to D (and V), and max denoting the function which selects the larger of two numbers.

These statistics show that our theoretically motivated expectations are supported. In particular, (1) SYNONYMOUS connectives have the least distributional divergence and EXCLUSIVE connectives the most, (2) CONT. SUBS. and HYPONYMOUS connectives have the greatest values for V, and (3) V shows the greatest sensitivity to the order of its arguments in the case of HYPONYMY.

The co-occurrence data were used to construct a Gaussian classifier, by assuming the values for D and V are generated by Gaussians.[2] First, normal functions were used to calculate the likelihood ratio of p and q being in the two relationships:

$$\frac{P(syn|data)}{P(hyp|data)} = \frac{P(syn)}{P(hyp)} \cdot \frac{P(data|syn)}{P(data|hyp)} \tag{8}$$

$$= 1 \cdot \frac{n(max(D_1, D_2); \mu_{syn}, \sigma_{syn})}{n(max(D_1, D_2); \mu_{hyp}, \sigma_{hyp})} \tag{9}$$

[2]KL divergence is right skewed, so a log-normal model was used to model D, whereas a normal model used for V.

where $n(x; \mu, \sigma)$ is the normal function with mean μ and standard deviation σ, and where μ_{syn}, for example, denotes the mean of the Gaussian model for SYNONYMY. Next the likelihood ratio for p and q was divided by that for p and q'. If this value was greater than 1, the model predicted p and q were SYNONYMS, otherwise HYPONYMS. The same technique was used for the second task.

4.3.2 Results

A leave-one-out cross validation procedure was used. For each triple $\langle p, q, q' \rangle$, the data concerning the pairs p, q and p, q' were held back, and the remaining data used to construct the models. The results are shown in Table 4. For comparison, a random baseline classifier achieves 50% accuracy.

The results demonstrate the utility of the new variance-based function V. The new variance-based function V is better than KL divergence at distinguishing HYPONYMY from SYNONYMY ($\chi^2 = 11.13, df = 1, p < 0.001$), although it performs worse on the coarser grained task. This is consistent with the expectations of Table 1. The two classifiers were also combined by making a naive Bayes assumption. This gave an accuracy of 76.1% on the first task, which is significantly better than just using KL divergence ($\chi^2 = 5.65, df = 1, p < 0.05$), and not significantly worse than using V. The combination's accuracy on the second task was 76.2%, which is about the same as using KL divergence. This shows that combining similarity- and variance-based measures can be useful can improve overall performance.

5 Conclusions

The concepts of lexical similarity and substitutability are of central importance to psychology, artificial intelligence and computational linguistics.

To our knowledge this is the first modelling study of how these concepts relate to lexical items involved in discourse-level phenomena. We found a three way correspondence between data sources of quite distinct types: distributional similarity scores obtained from lexical co-occurrence data, substitutability judgements made by linguists, and the similarity ratings of naive subjects.

The substitutability of lexical items is important for applications such as text simplification, where it can be desirable to paraphrase one discourse connective using another. Ultimately we would like to automatically predict substitutability for individual tokens. However predicting whether one connective can either a) always, b) sometimes or c) never be substituted for another is a step towards this goal. Our results demonstrate that these general substitutability relationships have empirical correlates.

We have introduced a novel variance-based function of two distributions which complements distributional similarity. We demonstrated the new function's utility in helping to predict the substitutability of connectives, and it can be expected to have wider applicability to lexical acquisition tasks. In particular, it is expected to be useful for learning relationships which cannot be characterised purely in terms of similarity, such as hyponymy. In future work we will analyse further the empirical properties of the new function, and investigate its applicability to learning relationships between other classes of lexical items such as nouns.

Acknowledgements

I would like to thank Mirella Lapata, Alex Lascarides, Alistair Knott, and the anonymous ACL reviewers for their helpful comments. This research was supported by EPSRC Grant GR/R40036/01 and a University of Sydney Travelling Scholarship.

References

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

James R. Curran and M. Moens. 2002. Improvements in automatic thesaurus extraction. In *Proceedings of the Workshop on Unsupervised Lexical Acquisition*, Philadelphia, USA.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Boston.

Brigitte Grote and Manfred Stede. 1998. Discourse marker choice in sentence planning. In Eduard Hovy, editor, *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 128–137, New Brunswick, New Jersey. Association for Computational Linguistics.

M. Halliday and R. Hasan. 1976. *Cohesion in English*. Longman.

Jerry A Hobbs. 1985. On the coherence and structure of discourse. Technical Report CSLI-85-37, Center for the Study of Language and Information, Stanford University.

Ben Hutchinson. 2004a. Acquiring the meaning of discourse markers. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 685–692.

Ben Hutchinson. 2004b. Mining the web for discourse markers. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 407–410, Lisbon, Portugal.

Ben Hutchinson. 2005. Modelling the similarity of discourse connectives. To appear in *Proceedings of the the 27th Annual Meeting of the Cognitive Science Society (CogSci2005)*.

Andrew Kehler. 2002. *Coherence, Reference and the Theory of Grammar*. CSLI publications.

Alistair Knott. 1996. *A data-driven methodology for motivating a set of coherence relations*. Ph.D. thesis, University of Edinburgh.

Mirella Lapata and Alex Lascarides. 2004. Inferring sentence-internal temporal relations. In *In Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics Annual Meeting*, Boston, MA.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of ACL 1999*.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, Philadelphia, PA.

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

Scott McDonald. 2000. *Environmental determinants of lexical processing effort*. Ph.D. thesis, University of Edinburgh.

George A. Miller and William G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

M. Moser and J. Moore. 1995. Using discourse analysis and automatic text generation to study discourse cue usage. In *Proceedings of the AAAI 1995 Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*.

Philip Resnik and Mona Diab. 2000. Measuring verb similarity. In *Proceedings of the Twenty Second Annual Meeting of the Cognitive Science Society*, Philadelphia, US, August.

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.

Advaith Siddharthan. 2003. Preserving discourse structure when simplifying text. In *Proceedings of the 2003 European Natural Language Generation Workshop*.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*, Sapporo, Japan, July.

Sholom M. Weiss and Casimir A. Kulikowski. 1991. *Computer systems that learn*. Morgan Kaufmann, San Mateo, CA.

Machine Learning for Coreference Resolution: From Local Classification to Global Ranking

Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
vince@hlt.utdallas.edu

Abstract

In this paper, we view coreference resolution as a problem of ranking candidate partitions generated by different coreference systems. We propose a set of partition-based features to learn a ranking model for distinguishing good and bad partitions. Our approach compares favorably to two state-of-the-art coreference systems when evaluated on three standard coreference data sets.

1 Introduction

Recent research in coreference resolution — the problem of determining which noun phrases (NPs) in a text or dialogue refer to which real-world entity — has exhibited a shift from knowledge-based approaches to data-driven approaches, yielding learning-based coreference systems that rival their hand-crafted counterparts in performance (e.g., Soon et al. (2001), Ng and Cardie (2002b), Strube et al. (2002), Yang et al. (2003), Luo et al. (2004)). The central idea behind the majority of these learning-based approaches is to recast coreference resolution as a binary classification task. Specifically, a classifier is first trained to determine whether two NPs in a document are co-referring or not. A separate clustering mechanism then coordinates the possibly contradictory pairwise coreference classification decisions and constructs a partition on the given set of NPs, with one cluster for each set of coreferent NPs.

Though reasonably successful, this "standard" approach is not as robust as one may think. First, design decisions such as the choice of the learning algorithm and the clustering procedure are apparently critical to system performance, but are often made in an ad-hoc and unprincipled manner that may be suboptimal from an empirical point of view.

Second, this approach makes no attempt to search through the space of possible partitions when given a set of NPs to be clustered, employing instead a greedy clustering procedure to construct a partition that may be far from optimal.

Another potential weakness of this approach concerns its inability to directly optimize for clustering-level accuracy: the coreference classifier is trained and optimized independently of the clustering procedure to be used, and hence improvements in classification accuracy do not guarantee corresponding improvements in clustering-level accuracy.

Our goal in this paper is to improve the robustness of the standard approach by addressing the above weaknesses. Specifically, we propose the following procedure for coreference resolution: given a set of NPs to be clustered, (1) use n pre-selected learning-based coreference systems to generate n candidate partitions of the NPs, and then (2) apply an *automatically acquired ranking model* to rank these candidate hypotheses, selecting the best one to be the final partition. The key features of this approach are:

Minimal human decision making. In contrast to the standard approach, our method obviates, to a large extent, the need to make tough or potentially suboptimal design decisions.[1] For instance, if we

[1]We still need to determine the n coreference systems to be employed in our framework, however. Fortunately, the choice of n is flexible, and can be as large as we want subject to the

cannot decide whether learner A is better to use than learner B in a coreference system, we can simply create two copies of the system with one employing A and the other B, and then add both into our pre-selected set of coreference systems.

Generation of multiple candidate partitions. Although an exhaustive search for the best partition is not computationally feasible even for a document with a moderate number of NPs, our approach explores a larger portion of the search space than the standard approach via generating multiple hypotheses, making it possible to find a potentially better partition of the NPs under consideration.

Optimization for clustering-level accuracy via ranking. As mentioned above, the standard approach trains and optimizes a coreference classifier without necessarily optimizing for clustering-level accuracy. In contrast, we attempt to optimize our ranking model with respect to the target coreference scoring function, essentially by training it in such a way that a higher scored candidate partition (according to the scoring function) would be assigned a higher rank (see Section 3.2 for details).

Perhaps even more importantly, our approach provides *a general framework* for coreference resolution. Instead of committing ourselves to a particular resolution method as in previous approaches, our framework makes it possible to leverage the strengths of different methods by allowing them to participate in the generation of candidate partitions.

We evaluate our approach on three standard coreference data sets using two different scoring metrics. In our experiments, our approach compares favorably to two state-of-the-art coreference systems adopting the standard machine learning approach, outperforming them by as much as 4–7% on the three data sets for one of the performance metrics.

2 Related Work

As mentioned before, our approach differs from the standard approach primarily by (1) explicitly learning a ranker and (2) optimizing for clustering-level accuracy. In this section we will focus on discussing related work along these two dimensions.

Ranking candidate partitions. Although we are not aware of any previous attempt on training a ranking model using global features of an NP partition, there is some related work on partition ranking where the score of a partition is computed via a heuristic function of the probabilities of its NP pairs being coreferent.[2] For instance, Harabagiu et al. (2001) introduce a greedy algorithm for finding the highest-scored partition by performing a beam search in the space of possible partitions. At each step of this search process, candidate partitions are ranked based on their heuristically computed scores.

Optimizing for clustering-level accuracy. Ng and Cardie (2002a) attempt to optimize their rule-based coreference classifier for clustering-level accuracy, essentially by finding a subset of the learned rules that performs the best on held-out data with respect to the target coreference scoring program. Strube and Müller (2003) propose a similar idea, but aim instead at finding a subset of the available features with which the resulting coreference classifier yields the best clustering-level accuracy on held-out data. To our knowledge, our work is the first attempt to optimize a ranker for clustering-level accuracy.

3 A Ranking Approach to Coreference

Our ranking approach operates by first dividing the available training texts into two disjoint subsets: a training subset and a held-out subset. More specifically, we first train each of our n pre-selected coreference systems on the documents in the training subset, and then use these resolvers to generate n candidate partitions for each text in the held-out subset from which a ranking model will be learned. Given a test text, we use our n coreference systems to create n candidate partitions as in training, and select the highest-ranked partition according to the ranking model to be the final partition.[3] The rest of this section describes how we select these n learning-based coreference systems and acquire the ranking model.

3.1 Selecting Coreference Systems

A learning-based coreference system can be defined by four elements: the *learning algorithm* used to train the coreference classifier, the *method of creating training instances* for the learner, the *feature set*

available computing resources.

[2]Examples of such scoring functions include the Dempster-Shafer rule (see Kehler (1997) and Bean and Riloff (2004)) and its variants (see Harabagiu et al. (2001) and Luo et al. (2004)).

[3]The ranking model breaks ties randomly.

used to represent a training or test instance, and the *clustering algorithm* used to coordinate the coreference classification decisions. Selecting a coreference system, then, is a matter of instantiating these elements with specific values.

Now we need to define the set of allowable values for each of these elements. In particular, we want to define them in such a way that the resulting coreference systems can potentially generate good candidate partitions. Given that machine learning approaches to the problem have been promising, our choices will be guided by previous learning-based coreference systems, as described below.

Training instance creation methods. A training instance represents two NPs, NP_i and NP_j, having a class value of COREFERENT or NOT COREFERENT depending on whether the NPs co-refer in the associated text. We consider three previously-proposed methods of creating training instances.

In **McCarthy and Lehnert's method**, a positive instance is created for each anaphoric NP paired with each of its antecedents, and a negative instance is created by pairing each NP with each of its preceding non-coreferent noun phrases. Hence, the number of instances created by this method is quadratic in the number of NPs in the associated text. The large number of instances can potentially make the training process inefficient.

In an attempt to reduce the training time, **Soon et al.'s method** creates a smaller number of training instances than McCarthy and Lehnert's. Specifically, a positive instance is created for each anaphoric NP, NP_j, and its closest antecedent, NP_i; and a negative instance is created for NP_j paired with each of the intervening NPs, $\text{NP}_{i+1}, \text{NP}_{i+2}, \ldots, \text{NP}_{j-1}$.

Unlike Soon et al., **Ng and Cardie's method** generates a positive instance for each anaphoric NP and its *most confident* antecedent. For a non-pronominal NP, the most confident antecedent is assumed to be its closest non-pronominal antecedent. For pronouns, the most confident antecedent is simply its closest preceding antecedent. Negative instances are generated as in Soon et al.'s method.

Feature sets. We employ two feature sets for representing an instance, as described below.

Soon et al.'s feature set consists of 12 surface-level features, each of which is computed based on one or both NPs involved in the instance. The features can be divided into four groups: lexical, grammatical, semantic, and positional. Space limitations preclude a description of these features. Details can be found in Soon et al. (2001).

Ng and Cardie expand Soon et al.'s feature set from 12 features to a deeper set of 53 to allow more complex NP string matching operations as well as finer-grained syntactic and semantic compatibility tests. See Ng and Cardie (2002b) for details.

Learning algorithms. We consider three learning algorithms, namely, the **C4.5** decision tree induction system (Quinlan, 1993), the **RIPPER** rule learning algorithm (Cohen, 1995), and **maximum entropy classification** (Berger et al., 1996). The classification model induced by each of these learners returns a number between 0 and 1 that indicates the likelihood that the two NPs under consideration are coreferent. In this work, NP pairs with class values above 0.5 are considered COREFERENT; otherwise the pair is considered NOT COREFERENT.

Clustering algorithms. We employ three clustering algorithms, as described below.

The **closest-first clustering** algorithm selects as the antecedent of NP_j its *closest* preceding coreferent NP. If no such NP exists, then NP_j is assumed to be non-anaphoric (i.e., no antecedent is selected).

On the other hand, the **best-first clustering** algorithm selects as the antecedent of NP_j the closest NP with the highest coreference likelihood value from its set of preceding coreferent NPs. If this set is empty, then no antecedent is selected for NP_j. Since the *most likely* antecedent is chosen for each NP, best-first clustering may produce partitions with higher precision than closest-first clustering.

Finally, in **aggressive-merge clustering**, each NP is merged with *all* of its preceding coreferent NPs. Since more merging occurs in comparison to the previous two algorithms, aggressive-merge clustering may yield partitions with higher recall.

Table 1 summarizes the previous work on coreference resolution that employs the learning algorithms, clustering algorithms, feature sets, and instance creation methods discussed above. With three learners, three training instance creation methods, two feature sets, and three clustering algorithms, we can produce 54 coreference systems in total.

Learning algorithm	Decision tree learners (C4.5/C5/CART)	Aone and Bennett (1995), McCarthy and Lehnert (1995), Soon et al. (2001), Strube et al. (2002), Strube and Müller (2003), Yang et al. (2003)
	RIPPER	Ng and Cardie (2002b)
	Maximum entropy	Kehler (1997), Morton (2000), Luo et al. (2004)
Instance creation method	McCarthy and Lehnert's	McCarthy and Lehnert (1995), Aone and Bennett (1995)
	Soon et al.'s	Soon et al. (2001), Strube et al. (2002), Iida et al. (2003)
	Ng and Cardie's	Ng and Cardie (2002b)
Feature set	Soon et al.'s	Soon et al. (2001)
	Ng and Cardie's	Ng and Cardie (2002b)
Clustering algorithm	Closest-fi rst	Soon et al. (2001), Strube et al. (2002)
	Best-fi rst	Aone and Bennett (1995), Ng and Cardie (2002b), Iida et al. (2003)
	Aggressive-merge	McCarthy and Lehnert (1995)

Table 1: Summary of the previous work on coreference resolution that employs the learning algorithms, the clustering algorithms, the feature sets, and the training instance creation methods discussed in Section 3.1.

3.2 Learning to Rank Candidate Partitions

We train an SVM-based ranker for ranking candidate partitions by means of Joachims' (2002) SVM^{light} package, with all the parameters set to their default values. To create training data, we first generate 54 candidate partitions for each text in the held-out subset as described above and then convert each partition into a training instance consisting of a set of *partition-based* features and *method-based* features.

Partition-based features are used to characterize a candidate partition and can be derived directly from the partition itself. Following previous work on using *global* features of candidate structures to learn a ranking model (Collins, 2002), the global (i.e., partition-based) features we consider here are simple functions of the *local* features that capture the relationship between NP pairs.

Specifically, we define our partition-based features in terms of the features in the Ng and Cardie (N&C) feature set (see Section 3.1) as follows. First, let us assume that f_i is the i-th nominal feature in N&C's feature set and v_{ij} is the j-th possible value of f_i. Next, for each i and j, we create two partition-based features, P_{ij0} and P_{ij1}. P_{ij0} is computed over the set of *coreferent* NP pairs (with respect to the candidate partition), denoting the probability of encountering $f_i = v_{ij}$ in this set when the pairs are represented as attribute-value vectors using N&C's features. On the other hand, P_{ij1} is computed over the set of *non-coreferent* NP pairs (with respect to the candidate partition), denoting the probability of encountering $f_i = v_{ij}$ in this set when the pairs are represented as attribute-value vectors using N&C's features. One partition-based feature, for instance, would denote the probability that two NPs residing in the same cluster have incompatible gender values. Intuitively, a good NP partition would have a low probability value for this feature. So, having these partition-based features can potentially help us distinguish good and bad candidate partitions.

Method-based features, on the other hand, are used to encode the identity of the coreference system that generated the candidate partition under consideration. Specifically, we have one method-based feature representing each pre-selected coreference system. The feature value is 1 if the corresponding coreference system generated the candidate partition and 0 otherwise. These features enable the learner to learn how to distinguish good and bad partitions based on the systems that generated them, and are particularly useful when some coreference systems perform consistently better than the others.

Now, we need to compute the "class value" for each training instance, which is a positive integer denoting the rank of the corresponding partition among the 54 candidates generated for the training document under consideration. Recall from the introduction that we want to train our ranking model so that higher scored partitions according to the target coreference scoring program are ranked higher. To this end, we compute the rank of each candidate partition as follows. First, we apply the target scoring program to score each candidate partition against the correct partition derived from the training text. We then assign rank i to the i-th lowest scored partition.[4] Effectively, the learning algorithm learns what a good partition is from the scoring program.

[4]Two partitions with the same score will have the same rank.

	Training Corpus		Test Corpus	
	# Docs	# Tokens	# Docs	# Tokens
BNEWS	216	67470	51	18357
NPAPER	76	71944	17	18174
NWIRE	130	85688	29	20528

Table 2: Statistics for the ACE corpus.

4 Evaluation

4.1 Experimental Setup

For evaluation purposes, we use the ACE (Automatic Content Extraction) coreference corpus, which is composed of three data sets created from three different news sources, namely, broadcast news (BNEWS), newspaper (NPAPER), and newswire (NWIRE).[5] Statistics of these data sets are shown in Table 2. In our experiments, we use the training texts to acquire coreference classifiers and evaluate the resulting systems on the test texts with respect to two commonly-used coreference scoring programs: the MUC scorer (Vilain et al., 1995) and the B-CUBED scorer (Bagga and Baldwin, 1998).

4.2 Results Using the MUC Scorer

Baseline systems. We employ as our baseline systems two existing coreference resolvers: our duplication of the Soon et al. (2001) system and the Ng and Cardie (2002b) system. Both resolvers adopt the standard machine learning approach and therefore can be characterized using the four elements discussed in Section 3.1. Specifically, Soon et al.'s system employs a decision tree learner to train a coreference classifier on instances created by Soon's method and represented by Soon's feature set, coordinating the classification decisions via closest-first clustering. Ng and Cardie's system, on the other hand, employs RIPPER to train a coreference classifier on instances created by N&C's method and represented by N&C's feature set, inducing a partition on the given NPs via best-first clustering.

The baseline results are shown in rows 1 and 2 of Table 3, where performance is reported in terms of recall, precision, and F-measure. As we can see, the N&C system outperforms the Duplicated Soon system by about 2-6% on the three ACE data sets.

[5] See `http://www.itl.nist.gov/iad/894.01/tests/ace` for details on the ACE research program.

Our approach. Recall that our approach uses labeled data to train both the coreference classifiers and the ranking model. To ensure a fair comparison of our approach with the baselines, we do not rely on additional labeled data for learning the ranker; instead, we use half of the training texts for training classifiers and the other half for ranking purposes. Results using our approach are shown in row 3 of Table 3. Our ranking model, when trained to optimize for F-measure using both partition-based features and method-based features, consistently provides substantial gains in F-measure over both baselines. In comparison to the stronger baseline (i.e., N&C), F-measure increases by 7.4, 7.2, and 4.6 for the BNEWS, NPAPER, and NWIRE data sets, respectively. Perhaps more encouragingly, gains in F-measure are accompanied by simultaneous increase in recall and precision for all three data sets.

Feature contribution. In an attempt to gain additional insight into the contribution of partition-based features and method-based features, we train our ranking model using each type of features in isolation. Results are shown in rows 4 and 5 of Table 3. For the NPAPER and NWIRE data sets, we still see gains in F-measure over both baseline systems when the model is trained using either type of features. The gains, however, are smaller than those observed when the two types of features are applied in combination. Perhaps surprisingly, the results for BNEWS do not exhibit the same trend as those for the other two data sets. Here, the method-based features alone are strongly predictive of good candidate partitions, yielding even slightly better performance than when both types of features are applied. Overall, however, these results seem to suggest that both partition-based and method-based features are important to learning a good ranking model.

Random ranking. An interesting question is: how much does supervised ranking help? If all of our candidate partitions are of very high quality, then ranking will not be particularly important because choosing any of these partitions may yield good results. To investigate this question, we apply a *random* ranking model, which randomly selects a candidate partition for each test text. Row 6 of Table 3 shows the results (averaged over five runs) when the random ranker is used in place of the supervised

		BNEWS			NPAPER			NWIRE		
	System Variation	R	P	F	R	P	F	R	P	F
1	Duplicated Soon et al. baseline	52.7	47.5	50.0	63.3	56.7	59.8	48.7	40.9	44.5
2	Ng and Cardie baseline	56.5	58.6	57.5	57.1	68.0	62.1	43.1	59.9	50.1
3	Ranking framework	62.2	67.9	64.9	67.4	71.4	69.3	50.1	60.3	54.7
4	Partition-based features only	54.5	55.5	55.0	66.3	63.0	64.7	50.7	51.2	51.0
5	Method-based features only	62.0	68.5	65.1	67.5	61.2	64.2	51.1	49.9	50.5
6	Random ranking model	48.6	54.8	51.5	57.4	63.3	60.2	40.3	44.3	42.2
7	Perfect ranking model	66.0	69.3	67.6	70.4	71.2	70.8	56.6	59.7	58.1

Table 3: Results for the three ACE data sets obtained via the MUC scoring program.

ranker. In comparison to the results in row 3, we see that the supervised ranker surpasses its random counterpart by about 9-13% in F-measure, implying that ranking plays an important role in our approach.

Perfect ranking. It would be informative to see whether our ranking model is performing at its upper limit, because further performance improvement beyond this point would require enlarging our set of candidate partitions. So, we apply a *perfect* ranking model, which uses an oracle to choose the best candidate partition for each test text. Results in row 7 of Table 3 indicate that our ranking model performs at about 1-3% below the perfect ranker, suggesting that we can further improve coreference performance by improving the ranking model.

4.3 Results Using the B-CUBED Scorer

Baseline systems. In contrast to the MUC results, the B-CUBED results for the two baseline systems are mixed (see rows 1 and 2 of Table 4). Specifically, while there is no clear winner for the NWIRE data set, N&C performs better on BNEWS but worse on NPAPER than the Duplicated Soon system.

Our approach. From row 3 of Table 4, we see that our approach achieves small but consistent improvements in F-measure over both baseline systems. In comparison to the better baseline, F-measure increases by 0.1, 1.1, and 2.0 for the BNEWS, NPAPER, and NWIRE data sets, respectively.

Feature contribution. Unlike the MUC results, using more features to train the ranking model does not always yield better performance with respect to the B-CUBED scorer (see rows 3-5 of Table 4). In particular, the best result for BNEWS is achieved using only method-based features, whereas the best result for NPAPER is obtained using only partition-based features. Nevertheless, since neither type of features offers consistently better performance than the other, it still seems desirable to apply the two types of features in combination to train the ranker.

Random ranking. Comparing rows 3 and 6 of Table 4, we see that the supervised ranker yields a nontrivial improvement of 2-3% in F-measure over the random ranker for the three data sets. Hence, ranking still plays an important role in our approach with respect to the B-CUBED scorer despite its modest performance gains over the two baseline systems.

Perfect ranking. Results in rows 3 and 7 of Table 4 indicate that the supervised ranker underperforms the perfect ranker by about 5% for BNEWS and 3% for both NPAPER and NWIRE in terms of F-measure, suggesting that the supervised ranker still has room for improvement. Moreover, by comparing rows 1-2 and 7 of Table 4, we can see that the perfect ranker outperforms the baselines by less than 5%. This is essentially an upper limit on how much our approach can improve upon the baselines given the current set of candidate partitions. In other words, the performance of our approach is limited in part by the quality of the candidate partitions, more so with B-CUBED than with the MUC scorer.

5 Discussion

Two questions naturally arise after examining the above results. First, which of the 54 coreference systems generally yield superior results? Second, why is the same set of candidate partitions scored so differently by the two scoring programs?

To address the first question, we take the 54 coreference systems that were trained on half of the available training texts (see Section 4) and apply them to the three ACE test data sets. Table 5 shows the best-performing resolver for each test set and scoring program combination. Interestingly, with respect to the

		BNEWS			NPAPER			NWIRE		
	System Variation	R	P	F	R	P	F	R	P	F
1	Duplicated Soon et al. baseline	53.4	78.4	63.5	58.0	75.4	65.6	56.0	75.3	64.2
2	Ng and Cardie baseline	59.9	72.3	65.5	61.8	64.9	63.3	62.3	66.7	64.4
3	Ranking framework	57.0	77.1	65.6	62.8	71.2	66.7	59.3	75.4	66.4
4	Partition-based features only	55.0	79.1	64.9	61.3	74.7	67.4	57.1	76.8	65.5
5	Method-based features only	63.1	69.8	65.8	58.4	75.2	65.8	58.9	75.5	66.1
6	Random ranking model	52.5	79.9	63.4	58.4	69.2	63.3	54.3	77.4	63.8
7	Perfect ranking model	64.5	76.7	70.0	61.3	79.1	69.1	63.2	76.2	69.1

Table 4: Results for the three ACE data sets obtained via the B-CUBED scoring program.

MUC scorer, the best performance on the three data sets is achieved by the same resolver. The results with respect to B-CUBED are mixed, however.

For each resolver shown in Table 5, we also compute the average rank of the partitions generated by the resolver for the corresponding test texts.[6] Intuitively, a resolver that consistently produces good partitions (relative to other candidate partitions) would achieve a low average rank. Hence, we can infer from the fairly high rank associated with the top B-CUBED resolvers that they do not perform consistently better than their counterparts.

Regarding our second question of why the same set of candidate partitions is scored differently by the two scoring programs, the reason can be attributed to two key algorithmic differences between these scorers. First, while the MUC scorer only rewards correct identification of coreferent links, B-CUBED additionally rewards successful recognition of non-coreference relationships. Second, the MUC scorer applies the same penalty to each erroneous merging decision, whereas B-CUBED penalizes erroneous merging decisions involving two large clusters more heavily than those involving two small clusters.

Both of the above differences can potentially cause B-CUBED to assign a narrower range of F-measure scores to each set of 54 candidate partitions than the MUC scorer, for the following reasons. First, our candidate partitions in general agree more on singleton clusters than on non-singleton clusters. Second, by employing a non-uniform penalty function B-CUBED effectively removes a bias inherent in the MUC scorer that leads to under-penalization of partitions in which entities are over-clustered.

Nevertheless, our B-CUBED results suggest that (1) despite its modest improvement over the baselines, our approach offers robust performance across the data sets; and (2) we could obtain better scores by improving the ranking model and expanding our set of candidate partitions, as elaborated below.

To improve the ranking model, we can potentially (1) design new features that better characterize a candidate partition (e.g., features that measure the size and the internal cohesion of a cluster), and (2) reserve more labeled data for training the model. In the latter case we may have less data for training coreference classifiers, but at the same time we can employ weakly supervised techniques to bootstrap the classifiers. Previous attempts on bootstrapping coreference classifiers have only been mildly successful (e.g., Müller et al. (2002)), and this is also an area that deserves further research.

To expand our set of candidate partitions, we can potentially incorporate more high-performing coreference systems into our framework, which is flexible enough to accommodate even those that adopt knowledge-based (e.g., Harabagiu et al. (2001)) and unsupervised approaches (e.g., Cardie and Wagstaff (1999), Bean and Riloff (2004)). Of course, we can also expand our pre-selected set of coreference systems via incorporating additional learning algorithms, clustering algorithms, and feature sets. Once again, we may use previous work to guide our choices. For instance, Iida et al. (2003) and Zelenko et al. (2004) have explored the use of SVM, voted perceptron, and logistic regression for training coreference classifiers. McCallum and Wellner (2003) and Zelenko et al. (2004) have employed graph-based partitioning algorithms such as correlation clustering (Bansal et al., 2002). Finally, Strube et al. (2002) and Iida et al. (2003) have proposed new edit-distance-based string-matching features and centering-based features, respectively.

[6]The rank of a partition is computed in the same way as in Section 3.2, except that we now adopt the common convention of assigning rank i to the i-th *highest* scored partition.

Test Set	Scoring Program	Average Rank	Coreference System			
			Instance Creation Method	Feature Set	Learner	Clustering Algorithm
BNEWS	MUC	7.2549	McCarthy and Lehnert's	Ng and Cardie's	C4.5	aggressive-merge
	BCUBED	16.9020	McCarthy and Lehnert's	Ng and Cardie's	C4.5	aggressive-merge
NPAPER	MUC	1.4706	McCarthy and Lehnert's	Ng and Cardie's	C4.5	aggressive-merge
	B-CUBED	9.3529	Soon et al.'s	Soon et al.'s	RIPPER	closest-fi rst
NWIRE	MUC	7.7241	McCarthy and Lehnert's	Ng and Cardie's	C4.5	aggressive-merge
	B-CUBED	13.1379	Ng and Cardie's	Ng and Cardie's	MaxEnt	closest-fi rst

Table 5: The coreference systems that achieved the highest F-measure scores for each test set and scorer combination. The average rank of the candidate partitions produced by each system for the corresponding test set is also shown.

Acknowledgments

We thank the three anonymous reviewers for their valuable comments on an earlier draft of the paper.

References

C. Aone and S. W. Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proc. of the ACL*, pages 122–129.

A. Bagga and B. Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proc. of COLING-ACL*, pages 79–85.

N. Bansal, A. Blum, and S. Chawla. 2002. Correlation clustering. In *Proc. of FOCS*, pages 238–247.

D. Bean and E. Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proc. of HLT/NAACL*, pages 297–304.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

C. Cardie and K. Wagstaff. 1999. Noun phrase coreference as clustering. In *Proc. of EMNLP/VLC*, pages 82–89.

W. Cohen. 1995. Fast effective rule induction. In *Proc. of ICML*, pages 115–123.

M. Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, pages 1–8.

S. Harabagiu, R. Bunescu, and S. Maiorano. 2001. Text and knowledge mining for coreference resolution. In *Proc. of NAACL*, pages 55–62.

R. Iida, K. Inui, H. Takamura, and Y. Matsumoto. 2003. Incorporating contextual cues in trainable models for coreference resolution. In *Proc. of the EACL Workshop on The Computational Treatment of Anaphora*.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proc. of KDD*, pages 133–142.

A. Kehler. 1997. Probabilistic coreference in information extraction. In *Proc. of EMNLP*, pages 163–173.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proc. of the ACL*, pages 136–143.

A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proc. of the IJCAI Workshop on Information Integration on the Web*.

J. McCarthy and W. Lehnert. 1995. Using decision trees for coreference resolution. In *Proc. of the IJCAI*, pages 1050–1055.

T. Morton. 2000. Coreference for NLP applications. In *Proc. of the ACL*.

C. M¨uller, S. Rapp, and M. Strube. 2002. Applying co-training to reference resolution. In *Proc. of the ACL*, pages 352–359.

V. Ng and C. Cardie. 2002a. Combining sample selection and error-driven pruning for machine learning of coreference rules. In *Proc. of EMNLP*, pages 55–62.

V. Ng and C. Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proc. of the ACL*, pages 104–111.

J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

W. M. Soon, H. T. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

M. Strube and C. M¨uller. 2003. A machine learning approach to pronoun resolution in spoken dialogue. In *Proc. of the ACL*, pages 168–175.

M. Strube, S. Rapp, and C. M¨uller. 2002. The influence of minimum edit distance on reference resolution. In *Proc. of EMNLP*, pages 312–319.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proc. of the Sixth Message Understanding Conference (MUC-6)*, pages 45–52.

X. Yang, G. D. Zhou, J. Su, and C. L. Tan. 2003. Coreference resolution using competitive learning approach. In *Proc. of the ACL*, pages 176–183.

D. Zelenko, C. Aone, and J. Tibbetts. 2004. Coreference resolution for information extraction. In *Proc. of the ACL Workshop on Reference Resolution and its Applications*, pages 9–16.

Improving Pronoun Resolution Using Statistics-Based Semantic Compatibility Information

Xiaofeng Yang[†‡] **Jian Su**[†] **Chew Lim Tan**[‡]

[†]Institute for Infocomm Research
21 Heng Mui Keng Terrace,
Singapore, 119613
{xiaofengy,sujian}@i2r.a-star.edu.sg

[‡] Department of Computer Science
National University of Singapore,
Singapore, 117543
{yangxiao,tancl}@comp.nus.edu.sg

Abstract

In this paper we focus on how to improve pronoun resolution using the statistics-based semantic compatibility information. We investigate two unexplored issues that influence the effectiveness of such information: statistics source and learning framework. Specifically, we for the first time propose to utilize the web and the twin-candidate model, in addition to the previous combination of the corpus and the single-candidate model, to compute and apply the semantic information. Our study shows that the semantic compatibility obtained from the web can be effectively incorporated in the twin-candidate learning model and significantly improve the resolution of neutral pronouns.

1 Introduction

Semantic compatibility is an important factor for pronoun resolution. Since pronouns, especially neutral pronouns, carry little semantics of their own, the compatibility between an anaphor and its antecedent candidate is commonly evaluated by examining the relationships between the candidate and the anaphor's context, based on the statistics that the corresponding predicate-argument tuples occur in a particular large corpus. Consider the example given in the work of Dagan and Itai (1990):

(1) *They know full well that companies held tax money aside for collection later on the basis that the government said* ***it***$_1$ *was going to collect* ***it***$_2$.

For anaphor *it*$_1$, the candidate *government* should have higher semantic compatibility than *money* because *government_collect* is supposed to occur more frequently than *money_collect* in a large corpus. A similar pattern could also be observed for *it*$_2$.

So far, the corpus-based semantic knowledge has been successfully employed in several anaphora resolution systems. Dagan and Itai (1990) proposed a heuristics-based approach to pronoun resolution. It determined the preference of candidates based on predicate-argument frequencies. Recently, Bean and Riloff (2004) presented an unsupervised approach to coreference resolution, which mined the co-referring NP pairs with similar predicate-arguments from a large corpus using a bootstrapping method.

However, the utility of the corpus-based semantics for pronoun resolution is often argued. Kehler et al. (2004), for example, explored the usage of the corpus-based statistics in supervised learning based systems, and found that such information did not produce apparent improvement for the overall pronoun resolution. Indeed, existing learning-based approaches to anaphor resolution have performed reasonably well using limited and shallow knowledge (e.g., Mitkov (1998), Soon et al. (2001), Strube and Muller (2003)). Could the relatively noisy semantic knowledge give us further system improvement?

In this paper we focus on improving pronominal anaphora resolution using automatically computed semantic compatibility information. We propose to enhance the utility of the statistics-based knowledge from two aspects:

Statistics source. Corpus-based knowledge usually suffers from data sparseness problem. That is, many predicate-argument tuples would be unseen even in a large corpus. A possible solution is the web. It is believed that the size of the web is thousands of times larger than normal large corpora, and the counts obtained from the web are highly correlated with the counts from large balanced corpora for predicate-argument bi-grams (Keller and Lapata, 2003). So far the web has been utilized in nominal anaphora resolution (Modjeska et al., 2003; Poesio et al., 2004) to determine the semantic relation between an anaphor and candidate pair. However, to our knowledge, using the web to help pronoun resolution still remains unexplored.

Learning framework. Commonly, the predicate-argument statistics is incorporated into anaphora resolution systems as a feature. What kind of learning framework is suitable for this feature? Previous approaches to anaphora resolution adopt the single-candidate model, in which the resolution is done on an anaphor and one candidate at a time (Soon et al., 2001; Ng and Cardie, 2002). However, as the purpose of the predicate-argument statistics is to evaluate the preference of the candidates in semantics, it is possible that the statistics-based semantic feature could be more effectively applied in the twin-candidate (Yang et al., 2003) that focusses on the preference relationships among candidates.

In our work we explore the acquisition of the semantic compatibility information from the corpus and the web, and the incorporation of such semantic information in the single-candidate model and the twin-candidate model. We systematically evaluate the combinations of different statistics sources and learning frameworks in terms of their effectiveness in helping the resolution. Results on the MUC data set show that for neutral pronoun resolution in which an anaphor has no specific semantic category, the web-based semantic information would be the most effective when applied in the twin-candidate model: Not only could such a system significantly improve the baseline without the semantic feature, it also outperforms the system with the combination of the corpus and the single-candidate model (by 11.5% success).

The rest of this paper is organized as follows. Section 2 describes the acquisition of the semantic compatibility information from the corpus and the web. Section 3 discusses the application of the statistics in the single-candidate and twin-candidate learning models. Section 4 gives the experimental results, and finally, Section 5 gives the conclusion.

2 Computing the Statistics-based Semantic Compatibility

In this section, we introduce in detail how to compute the semantic compatibility, using the predicate-argument statistics obtained from the corpus or the web.

2.1 Corpus-Based Semantic Compatibility

Three relationships, possessive-noun, subject-verb and verb-object, are considered in our work. Before resolution a large corpus is prepared. Documents in the corpus are processed by a shallow parser that could generate predicate-argument tuples of the above three relationships[1].

To reduce data sparseness, the following steps are applied in each resulting tuple, automatically:

- Only the nominal or verbal heads are retained.
- Each Named-Entity (NE) is replaced by a common noun which corresponds to the semantic category of the NE (e.g. "IBM" → "company") [2].
- All words are changed to their base morphologic forms (e.g. "companies → company").

During resolution, for an encountered anaphor, each of its antecedent candidates is substituted with the anaphor . According to the role and type of the anaphor in its context, a predicate-argument tuple is extracted and the above three steps for data-sparse reduction are applied. Consider the sentence (1), for example. The anaphors "it_1" and "it_2" indicate a subject_verb and verb_object relationship, respectively. Thus, the predicate-argument tuples for the two candidates "*government*" and "*money*" would be *(collect (subject government))* and *(collect (subject money))* for "it_1", and *(collect (object government))* and *(collect (object money))* for "it_2".

Each extracted tuple is searched in the prepared tuples set of the corpus, and the times the tuple occurs are calculated. For each candidate, its semantic

[1]The possessive-noun relationship involves the forms like "NP_2 *of* NP_1" and "NP_1*'s* NP_2".

[2]In our study, the semantic category of a NE is identified automatically by the pre-processing NE recognition component.

compatibility with the anaphor could be represented simply in terms of *frequency*

$$StatSem(candi, ana) = count(candi, ana) \quad (1)$$

where $count(candi, ana)$ is the count of the tuple formed by *candi* and *ana*, or alternatively, in terms of *conditional probability* ($P(candi, ana|candi)$), where the count of the tuple is divided by the count of the single candidate in the corpus. That is

$$StatSem(candi, ana) = \frac{count(candi, ana)}{count(candi)} \quad (2)$$

In this way, the statistics would not bias candidates that occur frequently in isolation.

2.2 Web-Based Semantic Compatibility

Unlike documents in normal corpora, web pages could not be preprocessed to generate the predicate-argument reserve. Instead, the predicate-argument statistics has to be obtained via a web search engine like Google and Altavista. For the three types of predicate-argument relationships, queries are constructed in the forms of "NP$_{candi}$ VP" (for subject-verb), "VP NP$_{candi}$" (for verb-object), and "NP$_{candi}$'s NP" or "NP of NP$_{candi}$" (for possessive-noun). Consider the following sentence:

(2) *Several experts suggested that IBM's accounting grew much more liberal since the mid 1980s as* ***its*** *business turned sour.*

For the pronoun "*its*" and the candidate "*IBM*", the two generated queries are "*business of IBM*" and "*IBM's business*".

To reduce data sparseness, in an initial query only the nominal or verbal heads are retained. Also, each NE is replaced by the corresponding common noun. (e.g, "*IBM's business*" $\rightarrow$ "*company's business*" and "*business of IBM*" $\rightarrow$ "*business of company*").

A set of inflected queries is generated by expanding a term into all its possible morphological forms. For example, in Sentence (1), "*collect money*" becomes "*collected*|*collecting*|... *money*", and in (2) "*business of company*" becomes "*business of company*|*companies*"). Besides, determiners are inserted for every noun. If the noun is the candidate under consideration, only the definite article *the* is inserted. For other nouns, instead, *a/an*, *the* and the empty determiners (for bare plurals) would be added (e.g., "*the*|*a business of the company*|*companies*").

Queries are submitted to a particular web search engine (Google in our study). All queries are performed as exact matching. Similar to the corpus-based statistics, the compatibility for each candidate and anaphor pair could be represented using either *frequency* (Eq. 1) or *probability* (Eq. 2) metric. In such a situation, $count(candi, ana)$ is the hit number of the inflected queries returned by the search engine, while $count(candi)$ is the hit number of the query formed with only the head of the candidate (i.e.,"*the* + *candi*").

3 Applying the Semantic Compatibility

In this section, we discuss how to incorporate the statistics-based semantic compatibility for pronoun resolution, in a machine learning framework.

3.1 The Single-Candidate Model

One way to utilize the semantic compatibility is to take it as a feature under the single-candidate learning model as employed by Ng and Cardie (2002).

In such a learning model, each training or testing instance takes the form of ***i***$\{$*C*, *ana*$\}$, where *ana* is the possible anaphor and *C* is its antecedent candidate. An instance is associated with a feature vector to describe their relationships.

During training, for each anaphor in a given text, a positive instance is created by pairing the anaphor and its closest antecedent. Also a set of negative instances is formed by pairing the anaphor and each of the intervening candidates. Based on the training instances, a binary classifier is generated using a certain learning algorithm, like C5 (Quinlan, 1993) in our work.

During resolution, given a new anaphor, a test instance is created for each candidate. This instance is presented to the classifier, which then returns a positive or negative result with a confidence value indicating the likelihood that they are co-referent. The candidate with the highest confidence value would be selected as the antecedent.

3.2 Features

In our study we only consider those domain-independent features that could be obtained with low

Feature	Description
DefNp	1 if the candidate is a definite NP; else 0
Pron	1 if the candidate is a pronoun; else 0
NE	1 if the candidate is a named entity; else 0
SameSent	1 if the candidate and the anaphor is in the same sentence; else 0
NearestNP	1 if the candidate is nearest to the anaphor; else 0
ParalStuct	1 if the candidate has an parallel structure with ana; else 0
FirstNP	1 if the candidate is the first NP in a sentence; else 0
Reflexive	1 if the anaphor is a reflexive pronoun; else 0
Type	Type of the anaphor (0: Single neuter pronoun; 1: Plural neuter pronoun; 2: Male personal pronoun; 3: Female personal pronoun)
StatSem*	the statistics-base semantic compatibility of the candidate
SemMag**	the semantic compatibility difference between two competing candidates

Table 1: Feature set for our pronoun resolution system(*ed feature is only for the single-candidate model while **ed feature is only for the twin-candidate mode)

computational cost but with high reliability. Table 1 summarizes the features with their respective possible values. The first three features represent the lexical properties of a candidate. The POS properties could indicate whether a candidate refers to a hearer-old entity that would have a higher preference to be selected as the antecedent (Strube, 1998). *SameSent* and *NearestNP* mark the distance relationships between an anaphor and the candidate, which would significantly affect the candidate selection (Hobbs, 1978). *FirstNP* aims to capture the salience of the candidate in the local discourse segment. *ParalStuct* marks whether a candidate and an anaphor have similar surrounding words, which is also a salience factor for the candidate evaluation (Mitkov, 1998).

Feature *StatSem* records the statistics-based semantic compatibility computed, from the corpus or the web, by either *frequency* or *probability* metric, as described in the previous section. If a candidate is a pronoun, this feature value would be set to that of its closest nominal antecedent.

As described, the semantic compatibility of a candidate is computed under the context of the current anaphor. Consider two occurrences of anaphors "... it_1 *collected* ..." and "... it_2 *said* ...". As "NP *collected*" should occur less frequently than "NP *said*", the candidates of it_1 would generally have predicate-argument statistics lower than those of it_2. That is, a positive instance for it_1 might bear a lower semantic feature value than a negative instance for it_2. The consequence is that the learning algorithm would think such a feature is not that "indicative" and reduce its salience in the resulting classifier.

One way to tackle this problem is to normalize the feature by the frequencies of the anaphor's context, e.g., "*count(collected)*" and "*count(said)*". This, however, would require extra calculation. In fact, as candidates of a specific anaphor share the same anaphor context, we can just normalize the semantic feature of a candidate by that of its competitor:

$$StatSem_N(C, ana) = \frac{StatSem(C, ana)}{\max\limits_{c_i \in candi\_set(ana)} StatSem(c_i, ana)}$$

The value ($0 \sim 1$) represents the rank of the semantic compatibility of the candidate *C* among $candi\_set(ana)$, the current candidates of ana.

3.3 The Twin-Candidate Model

Yang et al. (2003) proposed an alternative twin-candidate model for anaphora resolution task. The strength of such a model is that unlike the single-candidate model, it could capture the preference relationships between competing candidates. In the model, candidates for an anaphor are paired and features from two competing candidates are put together for consideration. This property could nicely deal with the above mentioned training problem of different anaphor contexts, because the semantic feature would be considered under the current candidate set only. In fact, as semantic compatibility is a preference-based factor for anaphor resolution, it would be incorporated in the twin-candidate model more naturally.

In the twin-candidate model, an instance takes a form like ***i***$\{C_1$, C_2, *ana*$\}$, where C_1 and C_2 are two candidates. We stipulate that C_2 should be closer to *ana* than C_1 in distance. The instance is labelled as "10" if C_1 the antecedent, or "01" if C_2 is.

During training, for each anaphor, we find its closest antecedent, C_{ante}. A set of "10" instances, ***i***$\{C_{ante}$, *C*, *ana*$\}$, is generated by pairing C_{ante} and each of the interning candidates *C*. Also a set of "01" instances, ***i***$\{C$, C_{ante}, *ana*$\}$, is created by pairing C_{ante} with each candidate before C_{ante} until another antecedent, if any, is reached.

The resulting pairwise classifier would return "10" or "01" indicating which candidate is preferred to the other. During resolution, candidates are paired one by one. The score of a candidate is the total number of the competitors that the candidate wins over. The candidate with the highest score would be selected as the antecedent.

Features The features for the twin-candidate model are similar to those for the single-candidate model except that a duplicate set of features has to be prepared for the additional candidate. Besides, a new feature, *SemMag*, is used in place of *StatSem* to represent the difference magnitude between the semantic compatibility of two candidates. Let $mag = StatSem(C_1, ana)/StatSem(C_2, ana)$, feature *SemMag* is defined as follows,

$$SemMag(C_1, C_2, ana) = \begin{cases} mag - 1 & : mag >= 1 \\ 1 - mag^{-1} & : mag < 1 \end{cases}$$

The positive or negative value marks the times that the statistics of C_1 is larger or smaller than C_2.

4 Evaluation and Discussion

4.1 Experiment Setup

In our study we were only concerned about the third-person pronoun resolution. With an attempt to examine the effectiveness of the semantic feature on different types of pronouns, the whole resolution was divided into neutral pronoun (*it* & *they*) resolution and personal pronoun (*he* & *she*) resolution.

The experiments were done on the newswire domain, using MUC corpus (Wall Street Journal articles). The training was done on 150 documents from MUC-6 coreference data set, while the testing was on the 50 formal-test documents of MUC-6 (30) and MUC-7 (20). Throughout the experiments, default learning parameters were applied to the C5 algorithm. The performance was evaluated based on *success*, the ratio of the number of correctly resolved anaphors over the total number of anaphors.

An input raw text was preprocessed automatically by a pipeline of NLP components. The noun phrase identification and the predicate-argument extraction were done based on the results of a chunk tagger, which was trained for the shared task of CoNLL-2000 and achieved 92% accuracy (Zhou et al., 2000). The recognition of NEs as well as their semantic categories was done by a HMM based NER, which was trained for the MUC NE task and obtained high F-scores of 96.9% (MUC-6) and 94.3% (MUC-7) (Zhou and Su, 2002).

For each anaphor, the markables occurring within the current and previous two sentences were taken as the initial candidates. Those with mismatched number and gender agreements were filtered from the candidate set. Also, pronouns or NEs that disagreed in person with the anaphor were removed in advance. For the training set, there are totally 645 neutral pronouns and 385 personal pronouns with non-empty candidate set, while for the testing set, the number is 245 and 197.

4.2 The Corpus and the Web

The corpus for the predicate-argument statistics computation was from the TIPSTER's Text Research Collection (v1994). Consisting of 173,252 Wall Street Journal articles from the year 1988 to 1992, the data set contained about 76 million words. The documents were preprocessed using the same POS tagging and NE-recognition components as in the pronoun resolution task. Cass (Abney, 1996), a robust chunker parser was then applied to generate the shallow parse trees, which resulted in 353,085 possessive-noun tuples, 759,997 verb-object tuples and 1,090,121 subject-verb tuples.

We examined the capacity of the web and the corpus in terms of zero-count ratio and count number. On average, among the predicate-argument tuples that have non-zero corpus-counts, above 93% have also non-zero web-counts. But the ratio is only around 40% contrariwise. And for the predicate-

		Neutral Pron		Personal Pron		Overall	
Learning Model	System	Corpus	Web	Corpus	Web	Corpus	Web
Single-Candidate	baseline	65.7		86.8		75.1	
	+frequency	67.3	69.9	86.8	86.8	76.0	76.9
	+normalized frequency	66.9	67.8	86.8	86.8	75.8	76.2
	+probability	65.7	65.7	86.8	86.8	75.1	75.1
	+normalized probability	67.7	70.6	86.8	86.8	76.2	77.8
Twin-Candidate	baseline	73.9		91.9		81.9	
	+frequency	76.7	**79.2**	91.4	91.9	83.3	**84.8**
	+probability	75.9	78.0	91.4	**92.4**	82.8	84.4

Table 2: The performance of different resolution systems

Relationship	N-Pron	P-Pron
Possessive-Noun	0.508	0.517
Verb-Object	0.503	0.526
Subject-Verb	0.619	0.676

Table 3: Correlation between web and corpus counts on the seen predicate-argument tuples

argument tuples that could be seen in both data sources, the count from the web is above 2000 times larger than that from the corpus.

Although much less sparse, the web counts are significantly noisier than the corpus count since no tagging, chunking and parsing could be carried out on the web pages. However, previous study (Keller and Lapata, 2003) reveals that the large amount of data available for the web counts could outweigh the noisy problems. In our study we also carried out a correlation analysis[3] to examine whether the counts from the web and the corpus are linearly related, on the predicate-argument tuples that can be seen in both data sources. From the results listed in Table 3, we observe moderately high correlation, with coefficients ranging from 0.5 to 0.7 around, between the counts from the web and the corpus, for both neutral pronoun (N-Pron) and personal pronoun (P-Pron) resolution tasks.

4.3 System Evaluation

Table 2 summarizes the performance of the systems with different combinations of statistics sources and learning frameworks. The systems without the semantic feature were used as the baseline. Under the single-candidate (SC) model, the baseline system obtains a success of 65.7% and 86.8% for neutral pronoun and personal pronoun resolution, respectively. By contrast, the twin-candidate (TC) model achieves a significantly ($p \leq 0.05$, by two-tailed t-test) higher success of 73.9% and 91.9%, respectively. Overall, for the whole pronoun resolution, the baseline system under the TC model yields a success 81.9%, 6.8% higher than SC does[4]. The performance is comparable to most state-of-the-art pronoun resolution systems on the same data set.

Web-based feature vs. Corpus-based feature The third column of the table lists the results using the web-based compatibility feature for neutral pronouns. Under both SC and TC models, incorporation of the web-based feature significantly boosts the performance of the baseline: For the best system in the SC model and the TC model, the success rate is improved significantly by around 4.9% and 5.3%, respectively. A similar pattern of improvement could be seen for the corpus-based semantic feature. However, the increase is not as large as using the web-based feature: Under the two learning models, the success rate of the best system with the corpus-based feature rises by up to 2.0% and 2.8% respectively, about 2.9% and 2.5% less than that of the counterpart systems with the web-based feature. The larger size and the better counts of the web against the corpus, as reported in Section 4.2,

[3]All the counts were log-transformed and the correlation coefficients were evaluated based on Pearsons' r.

[4]The improvement against SC is higher than that reported in (Yang et al., 2003). It should be because we now used 150 training documents rather than 30 ones as in the previous work. The TC model would benefit from larger training data set as it uses more features (more than double) than SC.

should contribute to the better performance.

Single-candidate model vs. Twin-Candidate model The difference between the SC and the TC model is obvious from the table. For the N-Pron and P-Pron resolution, the systems under TC could outperform the counterpart systems under SC by above 5% and 8% success, respectively. In addition, the utility of the statistics-based semantic feature is more salient under TC than under SC for N-Pron resolution: the best gains using the corpus-based and the web-based semantic features under TC are 2.9% and 5.3% respectively, higher than those under the SC model using either un-normalized semantic features (1.6% and 3.3%), or normalized semantic features (2.0% and 4.9%). Although under SC, the normalized semantic feature could result in a gain close to under TC, its utility is not stable: with metric *frequency*, using the normalized feature performs even worse than using the un-normalized one. These results not only affirm the claim by Yang et al. (2003) that the TC model is superior to the SC model for pronoun resolution, but also indicate that TC is more reliable than SC in applying the statistics-based semantic feature, for N-Pron resolution.

Web+TC vs. Other combinations The above analysis has exhibited the superiority of the web over the corpus, and the TC model over the SC model. The experimental results also reveal that using the the web-based semantic feature together with the TC model is able to further boost the resolution performance for neutral pronouns. The system with such a Web+TC combination could achieve a high success of 79.2%, defeating all the other possible combinations. Especially, it considerably outperforms (up to 11.5% success) the system with the Corpus+SC combination, which is commonly adopted in previous work (e.g., Kehler et al. (2004)).

Personal pronoun resolution vs. Neutral pronoun resolution Interestingly, the statistics-based semantic feature has no effect on the resolution of personal pronouns, as shown in the table 2. We found in the learned decision trees such a feature did not occur (SC) or only occurred in bottom nodes (TC). This should be because personal pronouns have strong restriction on the semantic category (i.e., *human*) of the candidates. A non-human candidate, even with a high predicate-argument statistics, could not be used as the antecedent (e.g. *company_said* in the sentence "*... the company ... he said ...*"). In fact, our analysis of the current data set reveals that most P-Prons refer back to a P-Pron or NE candidate whose semantic category (*human*) has been determined. That is, simply using features *NE* and *Pron* is sufficient to guarantee a high success, and thus the relatively weak semantic feature would not be taken in the learned decision tree for resolution.

Feature Group	Isolated	Combined
SemMag (Web-based)	61.2	61.2
Type+Reflexive	53.1	61.2
ParaStruct	53.1	61.2
Pron+DefNP+InDefNP+NE	57.1	67.8
NearestNP+SameSent	53.1	70.2
FirstNP	**65.3**	**79.2**

Table 4: Results of different feature groups under the TC model for N-pron resolution

```
SameSent_1 = 0:
:..SemMag > 0:
:  :..Pron_2 = 0: 10 (200/23)
:  :  Pron_2 = 1: ...
:  SemMag <= 0:
:  :..Pron_2 = 1: 01 (75/1)
:     Pron_2 = 0:
:     :..SemMag <= -28: 01 (110/19)
:        SemMag > -28: ...
SameSent_1 = 1:
:..SameSent_2 = 0: 01 (1655/49)
   SameSent_2 = 1:
   :..FirstNP_2 = 1: 01 (104/1)
      FirstNP_2 = 0:
      :..ParaStruct_2 = 1: 01 (3)
         ParaStruct_2 = 0:
         :..SemMag <= -151: 01 (27/2)
            SemMag > -151:...
```

Figure 1: Top portion of the decision tree learned under TC model for N-pron resolution (features ended with "_1" are for the first candidate C_1 and those with "_2" are for C_2.)

4.4 Feature Analysis

In our experiment we were also concerned about the importance of the web-based compatibility feature (using *frequency* metric) among the feature set. For this purpose, we divided the features into groups, and then trained and tested on one group at a time. Table 4 lists the feature groups and their respective results for N-Pron resolution under the TC model.

The second column is for the systems with only the current feature group, while the third column is with the features combined with the existing feature set. We see that used in isolation, the semantic compatibility feature is able to achieve a success up to 61% around, just 4% lower than the best indicative feature *FirstNP*. In combination with other features, the performance could be improved by as large as 18% as opposed to being used alone.

Figure 1 shows the top portion of the pruned decision tree for N-Pron resolution under the TC model. We could find that: (i) When comparing two candidates which occur in the same sentence as the anaphor, the web-based semantic feature would be examined in the first place, followed by the lexical property of the candidates. (ii) When two non-pronominal candidates are both in previous sentences before the anaphor, the web-based semantic feature is still required to be examined after *FirstNP* and *ParaStruct*. The decision tree further indicates that the web-based feature plays an important role in N-Pron resolution.

5 Conclusion

Our research focussed on improving pronoun resolution using the statistics-based semantic compatibility information. We explored two issues that affect the utility of the semantic information: statistics source and learning framework. Specifically, we proposed to utilize the web and the twin-candidate model, in addition to the common combination of the corpus and single-candidate model, to compute and apply the semantic information.

Our experiments systematically evaluated different combinations of statistics sources and learning models. The results on the newswire domain showed that the web-based semantic compatibility could be the most effectively incorporated in the twin-candidate model for the neutral pronoun resolution. While the utility is not obvious for personal pronoun resolution, we can still see the improvement on the overall performance. We believe that the semantic information under such a configuration would be even more effective on technical domains where neutral pronouns take the majority in the pronominal anaphors. Our future work would have a deep exploration on such domains.

References

S. Abney. 1996. Partial parsing via finite-state cascades. In *Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information*, pages 8–15.

D. Bean and E. Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proceedings of 2004 North American chapter of the Association for Computational Linguistics annual meeting*.

I. Dagan and A. Itai. 1990. Automatic processing of large corpora for the resolution of anahora references. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 330–332.

J. Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:339–352.

A. Kehler, D. Appelt, L. Taylor, and A. Simma. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of 2004 North American chapter of the Association for Computational Linguistics annual meeting*.

F. Keller and M. Lapata. 2003. Using the web to obtain freqencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Proceedings of the 17th Int. Conference on Computational Linguistics*, pages 869–875.

N. Modjeska, K. Markert, and M. Nissim. 2003. Using the web in machine learning for other-anaphora resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 176–183.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Philadelphia.

M. Poesio, R. Mehta, A. Maroudas, and J. Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of 42th Annual Meeting of the Association for Computational Linguistics*.

J. R. Quinlan. 1993. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers, San Francisco, CA.

W. Soon, H. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

M. Strube and C. Muller. 2003. A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 168–175, Japan.

M. Strube. 1998. Never look back: An alternative to centering. In *Proceedings of the 17th Int. Conference on Computational Linguistics and 36th Annual Meeting of ACL*, pages 1251–1257.

X. Yang, G. Zhou, J. Su, and C. Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Japan.

G. Zhou and J. Su. 2002. Named Entity recognition using a HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia.

G. Zhou, J. Su, and T. Tey. 2000. Hybrid text chunking. In *Proceedings of the 4th Conference on Computational Natural Language Learning*, pages 163–166, Lisbon, Portugal.

Coarse-to-fine n-best parsing and MaxEnt discriminative reranking

Eugene Charniak and Mark Johnson
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{mj|ec}@cs.brown.edu

Abstract

Discriminative reranking is one method for constructing high-performance statistical parsers (Collins, 2000). A discriminative reranker requires a source of candidate parses for each sentence. This paper describes a simple yet novel method for constructing sets of 50-best parses based on a coarse-to-fine generative parser (Charniak, 2000). This method generates 50-best lists that are of substantially higher quality than previously obtainable. We used these parses as the input to a MaxEnt reranker (Johnson et al., 1999; Riezler et al., 2002) that selects the best parse from the set of parses for each sentence, obtaining an f-score of 91.0% on sentences of length 100 or less.

1 Introduction

We describe a reranking parser which uses a regularized MaxEnt reranker to select the best parse from the 50-best parses returned by a generative parsing model. The 50-best parser is a probabilistic parser that on its own produces high quality parses; the maximum probability parse trees (according to the parser's model) have an f-score of 0.897 on section 23 of the Penn Treebank (Charniak, 2000), which is still state-of-the-art. However, the 50 best (i.e., the 50 highest probability) parses of a sentence often contain considerably better parses (in terms of f-score); this paper describes a 50-best parsing algorithm with an oracle f-score of 96.8 on the same data.

The reranker attempts to select the best parse for a sentence from the 50-best list of possible parses for the sentence. Because the reranker only has to consider a relatively small number of parses per sentences, it is not necessary to use dynamic programming, which permits the features to be essentially arbitrary functions of the parse trees. While our reranker does not achieve anything like the oracle f-score, the parses it selects do have an f-score of 91.0, which is considerably better than the maximum probability parses of the n-best parser.

In more detail, for each string s the n-best parsing algorithm described in section 2 returns the n highest probability parses $\mathcal{Y}(s) = \{y_1(s), \ldots, y_n(s)\}$ together with the probability $p(y)$ of each parse y according to the parser's probability model. The number n of parses was set to 50 for the experiments described here, but some simple sentences actually received fewer than 50 parses (so n is actually a function of s). Each yield or terminal string in the training, development and test data sets is mapped to such an n-best list of parse/probability pairs; the cross-validation scheme described in Collins (2000) was used to avoid training the n-best parser on the sentence it was being used to parse.

A feature extractor, described in section 3, is a vector of m functions $f = (f_1, \ldots, f_m)$, where each f_j maps a parse y to a real number $f_j(y)$, which is the value of the jth feature on y. So a feature extractor maps each y to a vector of feature values $f(y) = (f_1(y), \ldots, f_m(y))$.

Our reranking parser associates a parse with a score $v_\theta(y)$, which is a linear function of the feature values $f(y)$. That is, each feature f_j is associated with a weight θ_j, and the feature values and weights define the score $v_\theta(y)$ of each parse y as follows:

$$v_\theta(y) \ = \ \theta \cdot f(y) = \sum_{j=1}^{m} \theta_j f_j(y).$$

Given a string s, the reranking parser's output $\hat{y}(s)$ on string s is the highest scoring parse in the n-best parses $\mathcal{Y}(s)$ for s, i.e.,

$$\hat{y}(s) \ = \ \arg\max_{y \in \mathcal{Y}(s)} v_\theta(y).$$

The feature weight vector θ is estimated from the labelled training corpus as described in section 4. Because we use labelled training data we know the correct parse $y_\star(s)$ for each sentence s in the training data. The correct parse $y_\star(s)$ is not always a member of the n-best parser's output $\mathcal{Y}(s)$, but we can identify the parses $\mathcal{Y}_+(s)$ in $\mathcal{Y}(s)$ with the highest f-scores. Informally, the estimation procedure finds a weight vector θ that maximizes the score $v_\theta(y)$ of the parses $y \in \mathcal{Y}_+(s)$ relative to the scores of the other parses in $\mathcal{Y}(s)$, for each s in the training data.

2 Recovering the *n*-best parses using coarse-to-fine parsing

The major difficulty in n-best parsing, compared to 1-best parsing, is dynamic programming. For example, n-best parsing is straight-forward in best-first search or beam search approaches that do not use dynamic programming: to generate more than one parse, one simply allows the search mechanism to create successive versions to one's heart's content.

A good example of this is the Roark parser (Roark, 2001) which works left-to right through the sentence, and abjures dynamic programming in favor of a beam search, keeping some large number of possibilities to extend by adding the next word, and then re-pruning. At the end one has a beam-width's number of best parses (Roark, 2001).

The Collins parser (Collins, 1997) *does* use dynamic programming in its search. That is, whenever a constituent with the same history is generated a second time, it is discarded if its probability is lower than the original version. If the opposite is true, then the original is discarded. This is fine if one only wants the first-best, but obviously it does not directly enumerate the n-best parses.

However, Collins (Collins, 2000; Collins and Koo, in submission) has created an n-best version of his parser by turning off dynamic programming (see the user's guide to Bikel's re-implementation of Collins' parser, http://www.cis.upenn.edu/ dbikel/software.html#stat-parser). As with Roark's parser, it is necessary to add a beam-width constraint to make the search tractable. With a beam width of 1000 the parser returns something like a 50-best list (Collins, personal communication), but the actual number of parses returned for each sentences varies. However, turning off dynamic programming results in a loss in efficiency. Indeed, Collins's n-best list of parses for section 24 of the Penn tree-bank has some sentences with only a single parse, because the n-best parser could not find any parses.

Now there are two known ways to produce n-best parses while retaining the use of dynamic programming: the obvious way and the clever way.

The clever way is based upon an algorithm developed by Schwartz and Chow (1990). Recall the key insight in the Viterbi algorithm: in the optimal parse the parsing decisions at each of the choice points that determine a parse must be optimal, since otherwise one could find a better parse. This insight extends to n-best parsing as follows. Consider the second-best parse: if it is to differ from the best parse, then at least one of its parsing decisions must be suboptimal. In fact, all but one of the parsing decisions in second-best parse must be optimal, and the one suboptimal decision must be the second-best choice at that choice point. Further, the nth-best parse can only involve at most n suboptimal parsing decisions, and all but one of these must be involved in one of the second through the $n-1$th-best parses. Thus the basic idea behind this approach to n-best parsing is to first find the best parse, then find the second-best parse, then the third-best, and so on. The algorithm was originally described for hidden Markov models.

Since this first draft of this paper we have become aware of two PCFG implementations of this algorithm (Jimenez and Marzal, 2000; Huang and Chang, 2005). The first was tried on relatively small grammars, while the second was implemented on top of the Bikel re-implementation of the Collins parser (Bikel, 2004) and achieved oracle results for 50-best parses similar to those we report below.

Here, however, we describe how to find n-best parses in a more straight-forward fashion. Rather than storing a single best parse of each edge, one stores n of them. That is, when using dynamic programming, rather than throwing away a candidate if it scores less than the best, one keeps it if it is one of the top n analyses for this edge discovered so far. This is really very straight-forward. The problem is space. Dynamic programming parsing algorithms for PCFGs require $O(m^2)$ dynamic programming states, where m is the length of the sentence, so an n-best parsing algorithm requires $O(nm^2)$. However things get much worse when the grammar is bilexicalized. As shown by Eisner (Eisner and Satta, 1999) the dynamic programming algorithms for bilexicalized PCFGs require $O(m^3)$ states, so a n-best parser would require $O(nm^3)$ states. Things become worse still in a parser like the one described in Charniak (2000) because it conditions on (and hence splits the dynamic programming states according to) features of the grandparent node in addition to the parent, thus multiplying the number of possible dynamic programming states even more. Thus nobody has implemented this version.

There is, however, one particular feature of the Charniak parser that mitigates the space problem: it is a "coarse-to-fine" parser. By "coarse-to-fine" we mean that it first produces a crude version of the parse using coarse-grained dynamic programming states, and then builds fine-grained analyses by splitting the most promising of coarse-grained states.

A prime example of this idea is from Goodman (1997), who describes a method for producing a simple but crude approximate grammar of a standard context-free grammar. He parses a sentence using the approximate grammar, and the results are used to constrain the search for a parse with the full CFG. He finds that total parsing time is greatly reduced.

A somewhat different take on this paradigm is seen in the parser we use in this paper. Here the parser first creates a parse forest based upon a much less complex version of the complete grammar. In particular, it only looks at standard CFG features, the parent and neighbor labels. Because this grammar encodes relatively little state information, its dynamic programming states are relatively coarse and hence there are comparatively few of them, so it can be efficiently parsed using a standard dynamic programming bottom-up CFG parser. However, precisely because this first stage uses a grammar that ignores many important contextual features, the best parse it finds will not, in general, be the best parse according to the finer-grained second-stage grammar, so clearly we do not want to perform best-first parsing with this grammar. Instead, the output of the first stage is a polynomial-sized packed parse forest which records the left and right string positions for each local tree in the parses generated by this grammar. The edges in the packed parse forest are then pruned, to focus attention on the coarse-grained states that are likely to correspond to high-probability fine-grained states. The edges are then pruned according to their marginal probability conditioned on the string s being parsed as follows:

$$p(n^i_{j,k} \mid s) = \frac{\alpha(n^i_{j,k})\beta(n^i_{j,k})}{p(s)} \qquad (1)$$

Here $n^i_{j,k}$ is a constituent of type i spanning the words from j to k, $\alpha(n^i_{j,k})$ is the outside probability of this constituent, and $\beta(n^i_{j,k})$ is its inside probability. From parse forest both α and β can be computed in time proportional to the size of the compact forest. The parser then removes all constituents $n^i_{j,k}$ whose probability falls below some preset threshold. In the version of this parser available on the web, this threshold is on the order of 10^{-4}.

The unpruned edges are then exhaustively evaluated according to the fine-grained probabilistic model; in effect, each coarse-grained dynamic programming state is split into one or more fine-grained dynamic programming states. As noted above, the fine-grained model conditions on information that is not available in the coarse-grained model. This includes the lexical head of one's parents, the part of speech of this head, the parent's and grandparent's category labels, etc. The fine-grained states investigated by the parser are constrained to be refinements of the coarse-grained states, which drastically reduces the number of fine-grained states that need to be investigated.

It is certainly possible to do dynamic programming parsing directly with the fine-grained grammar, but precisely because the fine-grained grammar conditions on a wide variety of non-local contextual information there would be a very large number of different dynamic programming states, so direct dynamic programming parsing with the fine-grained grammar would be very expensive in terms of time and memory.

As the second stage parse evaluates all the remaining constituents in all of the contexts in which they appear (e.g., what are the possible grand-parent labels) it keeps track of the most probable expansion of the constituent in that context, and at the end is able to start at the root and piece together the overall best parse.

Now comes the easy part. To create a 50-best parser we simply change the fine-grained version of 1-best algorithm in accordance with the "obvious" scheme outlined earlier in this section. The first, coarse-grained, pass is not changed, but the second, fine-grained, pass keeps the n-best possibilities at each dynamic programming state, rather than keeping just first best. When combining two constituents to form a larger constituent, we keep the best 50 of the 2500 possibilities they offer. Naturally, if we keep each 50-best list sorted, we do nothing like 2500 operations.

The experimental question is whether, in practice, the coarse-to-fine architecture keeps the number of dynamic programming states sufficiently low that space considerations do not defeat us.

The answer seems to be yes. We ran the algorithm on section 24 of the Penn WSJ tree-bank using the default pruning settings mentioned above. Table 1 shows how the number of fine-grained dynamic programming states increases as a function of sentence length for the sentences in section 24 of the Treebank. There are no sentences of length greater than 69 in this section. Columns two to four show the number of sentences in each bucket, their average length, and the average number of fine-grained dynamic programming structures per sentence. The final column gives the value of the function $100 * L^{1.5}$ where L is the average length of sentences in the bucket. Except for bucket 6, which is abnormally low, it seems that this add-hoc function tracks the number of structures quite well. Thus the number of dynamic programming states does not grow as L^2, much less as L^3.

To put the number of these structures per sentence in perspective, consider the size of such structures. Each one must contain a probability, the nonterminal label of the structure, and a vector of pointers to it's children (an average parent has slightly more than two children). If one were concerned about every byte this could be made quite small. In our implementation probably the biggest factor is the STL overhead on vectors. If we figure we are using, say, 25 bytes per structure, the total space required is only 1.25Mb even for 50,000 dynamic programming states, so it is clearly not worth worrying about the memory required.

Len	Num sents	Av sen length	Av strs per sent	$100 * L^{1.5}$
0–9	225	6.04	1167	1484
10–19	725	15.0	4246	5808
20–29	795	24.2	9357	11974
30–39	465	33.8	15893	19654
40–49	162	43.2	21015	28440
50–59	35	52.8	30670	38366
60–69	9	62.8	23405	49740

Table 1: Number of structures created as a function of sentence length

n	1	2	10	25	50
f-score	0.897	0.914	0.948	0.960	0.968

Table 2: Oracle f-score as a function of number n of n-best parses

The resulting n-bests are quite good, as shown in Table 2. (The results are for all sentences of section 23 of the WSJ tree-bank of length ≤ 100.) From the 1-best result we see that the base accuracy of the parser is 89.7%.[1] 2-best and 10-best show dramatic oracle-rate improvements. After that things start to slow down, and we achieve an oracle rate of 0.968 at 50-best. To put this in perspective, Roark (Roark, 2001) reports oracle results of 0.941 (with the same experimental setup) using his parser to return a variable number of parses. For the case cited his parser returns, on average, 70 parses per sentence.

Finally, we note that 50-best parsing is only a fac-

[1]Charniak in (Charniak, 2000) cites an accuracy of 89.5%. Fixing a few very small bugs discovered by users of the parser accounts for the difference.

tor of two or three slower than 1-best.

3 Features for reranking parses

This section describes how each parse y is mapped to a feature vector $f(y) = (f_1(y), \ldots, f_m(y))$. Each feature f_j is a function that maps a parse to a real number. The first feature $f_1(y) = \log p(y)$ is the logarithm of the parse probability p according to the n-best parser model. The other features are integer valued; informally, each feature is associated with a particular configuration, and the feature's value $f_j(y)$ is the number of times that the configuration that f_j indicates. For example, the feature $f_{\text{eat pizza}}(y)$ counts the number of times that a phrase in y headed by *eat* has a complement phrase headed by *pizza*.

Features belong to feature schema, which are abstract schema from which specific features are instantiated. For example, the feature $f_{\text{eat pizza}}$ is an instance of the "Heads" schema. Feature schema are often parameterized in various ways. For example, the "Heads" schema is parameterized by the type of heads that the feature schema identifies. Following Grimshaw (1997), we associate each phrase with a lexical head and a function head. For example, the lexical head of an NP is a noun while the functional head of an NP is a determiner, and the lexical head of a VP is a main verb while the functional head of VP is an auxiliary verb.

We experimented with various kinds of feature selection, and found that a simple count threshold performs as well as any of the methods we tried. Specifically, we ignored all features that did not vary on the parses of at least t sentences, where t is the count threshold. In the experiments described below $t = 5$, though we also experimented with $t = 2$.

The rest of this section outlines the feature schemata used in the experiments below. These feature schemata used here were developed using the n-best parses provided to us by Michael Collins approximately a year before the n-best parser described here was developed. We used the division into preliminary training and preliminary development data sets described in Collins (2000) while experimenting with feature schemata; i.e., the first 36,000 sentences of sections 2–20 were used as preliminary training data, and the remaining sentences of sections 20 and 21 were used as preliminary development data. It is worth noting that developing feature schemata is much more of an art than a science, as adding or deleting a single schema usually does not have a significant effect on performance, yet the overall impact of many well-chosen schemata can be dramatic.

Using the 50-best parser output described here, there are 1,148,697 features that meet the count threshold of at least 5 on the main training data (i.e., Penn treebank sections 2–21). We list each feature schema's name, followed by the number of features in that schema with a count of at least 5, together with a brief description of the instances of the schema and the schema's parameters.

CoPar (10) The instances of this schema indicate conjunct parallelism at various different depths. For example, conjuncts which have the same label are parallel at depth 0, conjuncts with the same label and whose children have the same label are parallel at depth 1, etc.

CoLenPar (22) The instances of this schema indicate the binned difference in length (in terms of number of preterminals dominated) in adjacent conjuncts in the same coordinated structures, conjoined with a boolean flag that indicates whether the pair is final in the coordinated phrase.

RightBranch (2) This schema enables the reranker to prefer right-branching trees. One instance of this schema returns the number of nonterminal nodes that lie on the path from the root node to the right-most non-punctuation preterminal node, and the other instance of this schema counts the number of the other nonterminal nodes in the parse tree.

Heavy (1049) This schema classifies nodes by their category, their binned length (i.e., the number of preterminals they dominate), whether they are at the end of the sentence and whether they are followed by punctuation.

Neighbours (38,245) This schema classifies nodes by their category, their binned length, and the part of speech categories of the ℓ_1 preterminals to the node's left and the ℓ_2 preterminals to the node's right. ℓ_1 and ℓ_2 are parameters of this schema; here $\ell_1 = 1$ or $\ell_1 = 2$ and $\ell_2 = 1$.

Rule (271,655) The instances of this schema are local trees, annotated with varying amounts of contextual information controlled by the schema's parameters. This schema was inspired by a similar schema in Collins and Koo (in submission). The parameters to this schema control whether nodes are annotated with their preterminal heads, their terminal heads and their ancestors' categories. An additional parameter controls whether the feature is specialized to embedded or non-embedded clauses, which roughly corresponds to Emonds' "non-root" and "root" contexts (Emonds, 1976).

NGram (54,567) The instances of this schema are ℓ-tuples of adjacent children nodes of the same parent. This schema was inspired by a similar schema in Collins and Koo (in submission). This schema has the same parameters as the Rule schema, plus the length ℓ of the tuples of children ($\ell = 2$ here).

Heads (208,599) The instances of this schema are tuples of head-to-head dependencies, as mentioned above. The category of the node that is the least common ancestor of the head and the dependent is included in the instance (this provides a crude distinction between different classes of arguments). The parameters of this schema are whether the heads involved are lexical or functional heads, the number of heads in an instance, and whether the lexical item or just the head's part of speech are included in the instance.

LexFunHeads (2,299) The instances of this feature are the pairs of parts of speech of the lexical head and the functional head of nodes in parse trees.

WProj (158,771) The instances of this schema are preterminals together with the categories of ℓ of their closest maximal projection ancestors. The parameters of this schema control the number ℓ of maximal projections, and whether the preterminals and the ancestors are lexicalized.

Word (49,097) The instances of this schema are lexical items together with the categories of ℓ of their immediate ancestor nodes, where ℓ is a schema parameter ($\ell = 2$ or $\ell = 3$ here). This feature was inspired by a similar feature in Klein and Manning (2003).

HeadTree (72,171) The instances of this schema are tree fragments consisting of the local trees consisting of the projections of a preterminal node and the siblings of such projections. This schema is parameterized by the head type (lexical or functional) used to determine the projections of a preterminal, and whether the head preterminal is lexicalized.

NGramTree (291,909) The instances of this schema are subtrees rooted in the least common ancestor of ℓ contiguous preterminal nodes. This schema is parameterized by the number ℓ of contiguous preterminals ($\ell = 2$ or $\ell = 3$ here) and whether these preterminals are lexicalized.

4 Estimating feature weights

This section explains how we estimate the feature weights $\theta = (\theta_1, \ldots, \theta_m)$ for the feature functions $f = (f_1, \ldots, f_m)$. We use a MaxEnt estimator to find the feature weights $\hat{\theta}$, where L is the loss function and R is a regularization penalty term:

$$\hat{\theta} = \arg\min_{\theta} L_D(\theta) + R(\theta).$$

The training data $D = (s_1, \ldots, s_{n'})$ is a sequence of sentences and their correct parses $y_\star(s_1), \ldots, y_\star(s_n)$. We used the 20-fold cross-validation technique described in Collins (2000) to compute the n-best parses $\mathcal{Y}(s)$ for each sentence s in D. In general the correct parse $y_\star(s)$ is not a member of $\mathcal{Y}(s)$, so instead we train the reranker to identify one of the best parses $\mathcal{Y}_+(s) = \arg\max_{y \in \mathcal{Y}(s)} F_{y_\star(s)}(y)$ in the n-best parser's output, where $F_{y_\star}(y)$ is the Parseval f-score of y evaluated with respect to $y_\star$.

Because there may not be a unique best parse for each sentence (i.e., $|\mathcal{Y}_+(s)| > 1$ for some sentences s) we used the variant of MaxEnt described in Riezler et al. (2002) for partially labelled training data.

Recall the standard MaxEnt conditional probability model for a parse $y \in \mathcal{Y}$:

$$\begin{aligned} \mathrm{P}_\theta(y|\mathcal{Y}) &= \frac{\exp v_\theta(y)}{\sum_{y' \in \mathcal{Y}} \exp v_\theta(y')}, \text{where} \\ v_\theta(y) &= \theta \cdot f(y) \; = \; \sum_{j=1}^{m} \theta_j f_j(y). \end{aligned}$$

The loss function L_D proposed in Riezler et al. (2002) is just the negative log conditional likelihood of the best parses $\mathcal{Y}_+(s)$ relative to the n-best parser output $\mathcal{Y}(s)$:

$$\begin{aligned} L_D(\theta) &= -\sum_{i=1}^{n'} \log \mathrm{P}_\theta(\mathcal{Y}_+(s_i)|\mathcal{Y}(s_i)), \text{where} \\ \mathrm{P}_\theta(\mathcal{Y}_+|\mathcal{Y}) &= \sum_{y \in \mathcal{Y}_+} \mathrm{P}_\theta(y|\mathcal{Y}) \end{aligned}$$

The partial derivatives of this loss function, which are required by the numerical estimation procedure, are:

$$\begin{aligned} \frac{\partial L_D}{\theta_j} &= \sum_{i=1}^{n'} \mathrm{E}_\theta[f_j|\mathcal{Y}(s_i)] - E_\theta[f_j|\mathcal{Y}_+(s_i)] \\ \mathrm{E}_\theta[f|\mathcal{Y}] &= \sum_{y \in \mathcal{Y}} f(y) \mathrm{P}_\theta(y|\mathcal{Y}) \end{aligned}$$

In the experiments reported here, we used a Gaussian or quadratic regularizer $R(w) = c \sum_{j=1}^{m} w_j^2$, where c is an adjustable parameter that controls the amount of regularization, chosen to optimize the reranker's f-score on the development set (section 24 of the treebank).

We used the Limited Memory Variable Metric optimization algorithm from the PETSc/TAO optimization toolkit (Benson et al., 2004) to find the optimal feature weights $\hat{\theta}$ because this method seems substantially faster than comparable methods (Malouf, 2002). The PETSc/TAO toolkit provides a variety of other optimization algorithms and flags for controlling convergence, but preliminary experiments on the Collins' trees with different algorithms and early stopping did not show any performance improvements, so we used the default PETSc/TAO setting for our experiments here.

5 Experimental results

We evaluated the performance of our reranking parser using the standard PARSEVAL metrics. We trained the n-best parser on sections 2–21 of the Penn Treebank, and used section 24 as development data to tune the mixing parameters of the smoothing model. Similarly, we trained the feature weights θ with the MaxEnt reranker on sections 2–21, and adjusted the regularizer constant c to maximize the f-score on section 24 of the treebank. We did this both on the trees supplied to us by Michael Collins, and on the output of the n-best parser described in this paper. The results are presented in Table 3. The n-best parser's most probable parses are already of state-of-the-art quality, but the reranker further improves the f-score.

n-best trees	**f-score**
New	0.9102
Collins	0.9037

Table 3: Results on new n-best trees and Collins n-best trees, with weights estimated from sections 2–21 and the regularizer constant c adjusted for optimal f-score on section 24 and evaluated on sentences of length less than 100 in section 23.

6 Conclusion

This paper has described a dynamic programming n-best parsing algorithm that utilizes a heuristic coarse-to-fine refinement of parses. Because the coarse-to-fine approach prunes the set of possible parse edges beforehand, a simple approach which enumerates the n-best analyses of each parse edge is not only practical but quite efficient.

We use the 50-best parses produced by this algorithm as input to a MaxEnt discriminative reranker. The reranker selects the best parse from this set of parses using a wide variety of features. The system we described here has an f-score of 0.91 when trained and tested using the standard PARSEVAL framework.

This result is only slightly higher than the highest reported result for this test-set, Bod's (.907) (Bod, 2003). More to the point, however, is that the system we describe is reasonably efficient so it can be used for the kind of routine parsing currently being handled by the Charniak or Collins parsers. A 91.0 f-score represents a 13% reduction in f-measure error over the best of these parsers.[2] Both the 50-best parser, and the reranking parser can be found at ftp://ftp.cs.brown.edu/pub/nlparser/, named parser and reranker respectively.

Acknowledgements We would like to thanks Michael Collins for the use of his data and many helpful comments, and Liang Huang for providing an early draft of his paper and very useful comments on our paper. Finally thanks to the National Science Foundation for its support (NSF IIS-0112432, NSF 9721276, and NSF DMS-0074276).

References

Steve Benson, Lois Curfman McInnes, Jorge J. Mor, and Jason Sarich. 2004. Tao users manual. Technical Report ANL/MCS-TM-242-Revision 1.6, Argonne National Laboratory.

Daniel M. Bikel. 2004. Intricacies of collins parsing model. *Computational Linguistics*, 30(4).

Rens Bod. 2003. An efficient implementation of an new dop model. In *Proceedings of the European Chapter of the Association for Computational Linguists*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *The Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

Michael Collins and Terry Koo. in submission. Discriminative reranking for natural language parsing. Technical report, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, San Francisco. Morgan Kaufmann.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML 2000)*, pages 175–182, Stanford, California.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–464.

Joseph Emonds. 1976. *A Transformational Approach to English Syntax: Root, Structure-Preserving and Local Transformations.* Academic Press, New York, NY.

Joshua Goodman. 1997. Global thresholding and multiple-pass parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 1997)*.

Jane Grimshaw. 1997. Projection, heads, and optimality. *Linguistic Inquiry*, 28(3):373–422.

Liang Huang and David Chang. 2005. Better k-best parsing. Technical Report MS-CIS-05-08, Department of Computer Science, University of Pennsylvania.

Victor M. Jimenez and Andres Marzal. 2000. Computation of the n best parse trees for weighted and stochastic context-free grammars. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*. Springer LNCS 1876.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic “unification-based” grammars. In *The Proceedings of the 37th Annual Conference of the Association for Computational Linguistics*, pages 535–541, San Francisco. Morgan Kaufmann.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pages 49–55.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. III Maxwell, 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. Morgan Kaufmann.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

R. Schwartz and Y.L. Chow. 1990. The n-best algorithm: An efficient and exact procedure for finding the n most likely sentence hypotheses. In *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal, Processing*, pages 81–84.

[2]This probably underestimates the actual improvement. There are no currently accepted figures for inter-annotater agreement on Penn WSJ, but it is no doubt well short of 100%. If we take 97% as a reasonable estimate of the the upper bound on tree-bank accuracy, we are instead talking about an 18% error reduction.

Data-Defined Kernels for Parse Reranking Derived from Probabilistic Models

James Henderson
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, United Kingdom
james.henderson@ed.ac.uk

Ivan Titov
Department of Computer Science
University of Geneva
24, rue Général Dufour
CH-1211 Genève 4, Switzerland
ivan.titov@cui.unige.ch

Abstract

Previous research applying kernel methods to natural language parsing have focussed on proposing kernels over parse trees, which are hand-crafted based on domain knowledge and computational considerations. In this paper we propose a method for defining kernels in terms of a probabilistic model of parsing. This model is then trained, so that the parameters of the probabilistic model reflect the generalizations in the training data. The method we propose then uses these trained parameters to define a kernel for reranking parse trees. In experiments, we use a neural network based statistical parser as the probabilistic model, and use the resulting kernel with the Voted Perceptron algorithm to rerank the top 20 parses from the probabilistic model. This method achieves a significant improvement over the accuracy of the probabilistic model.

1 Introduction

Kernel methods have been shown to be very effective in many machine learning problems. They have the advantage that learning can try to optimize measures related directly to expected testing performance (i.e. "large margin" methods), rather than the probabilistic measures used in statistical models, which are only indirectly related to expected testing performance. Work on kernel methods in natural language has focussed on the definition of appropriate kernels for natural language tasks. In particular, most of the work on parsing with kernel methods has focussed on kernels over parse trees (Collins and Duffy, 2002; Shen and Joshi, 2003; Shen et al., 2003; Collins and Roark, 2004). These kernels have all been hand-crafted to try reflect properties of parse trees which are relevant to discriminating correct parse trees from incorrect ones, while at the same time maintaining the tractability of learning.

Some work in machine learning has taken an alternative approach to defining kernels, where the kernel is derived from a probabilistic model of the task (Jaakkola and Haussler, 1998; Tsuda et al., 2002). This way of defining kernels has two advantages. First, linguistic knowledge about parsing is reflected in the design of the probabilistic model, not directly in the kernel. Designing probabilistic models to reflect linguistic knowledge is a process which is currently well understood, both in terms of reflecting generalizations and controlling computational cost. Because many NLP problems are unbounded in size and complexity, it is hard to specify all possible relevant kernel features without having so many features that the computations become intractable and/or the data becomes too sparse.[1] Second, the kernel is defined using the trained parameters of the probabilistic model. Thus the kernel is in part determined by the training data, and is automatically tailored to reflect properties of parse trees which are relevant to parsing.

[1]For example, see (Henderson, 2004) for a discussion of why generative models are better than models parameterized to estimate the a posteriori probability directly.

In this paper, we propose a new method for deriving a kernel from a probabilistic model which is specifically tailored to reranking tasks, and we apply this method to natural language parsing. For the probabilistic model, we use a state-of-the-art neural network based statistical parser (Henderson, 2003). The resulting kernel is then used with the Voted Perceptron algorithm (Freund and Schapire, 1998) to reranking the top 20 parses from the probabilistic model. This method achieves a significant improvement over the accuracy of the probabilistic model alone.

2 Kernels Derived from Probabilistic Models

In recent years, several methods have been proposed for constructing kernels from trained probabilistic models. As usual, these kernels are then used with linear classifiers to learn the desired task. As well as some empirical successes, these methods are motivated by theoretical results which suggest we should expect some improvement with these classifiers over the classifier which chooses the most probable answer according to the probabilistic model (i.e. the maximum a posteriori (MAP) classifier). There is guaranteed to be a linear classifier for the derived kernel which performs at least as well as the MAP classifier for the probabilistic model. So, assuming a large-margin classifier can optimize a more appropriate criteria than the posterior probability, we should expect the derived kernel's classifier to perform better than the probabilistic model's classifier, although empirical results on a given task are never guaranteed.

In this section, we first present two previous kernels and then propose a new kernel specifically for reranking tasks. In each of these discussions we need to characterize the parsing problem as a classification task. Parsing can be regarded as a mapping from an input space of sentences $x \in \mathcal{X}$ to a structured output space of parse trees $y \in \mathcal{Y}$. On the basis of training sentences, we learn a discriminant function $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$. The parse tree y with the largest value for this discriminant function $F(x, y)$ is the output parse tree for the sentence x. We focus on the linear discriminant functions:

$$F_w(x, y) = <w, \phi(x, y)>,$$

where $\phi(x, y)$ is a feature vector for the sentence-tree pair, w is a parameter vector for the discriminant function, and $<a, b>$ is the inner product of vectors a and b. In the remainder of this section, we will characterize the kernel methods we consider in terms of the feature extractor $\phi(x, y)$.

2.1 Fisher Kernels

The Fisher kernel (Jaakkola and Haussler, 1998) is one of the best known kernels belonging to the class of probability model based kernels. Given a generative model of $P(z|\hat{\theta})$ with smooth parameterization, the Fisher score of an example z is a vector of partial derivatives of the log-likelihood of the example with respect to the model parameters:

$$\phi_{\hat{\theta}}(z) = (\frac{\partial \log P(z|\hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial \log P(z|\hat{\theta})}{\partial \theta_l}).$$

This score can be regarded as specifying how the model should be changed in order to maximize the likelihood of the example z. Then we can define the similarity between data points as the inner product of the corresponding Fisher scores. This kernel is often referred to as the practical Fisher kernel. The theoretical Fisher kernel depends on the Fisher information matrix, which is not feasible to compute for most practical tasks and is usually omitted.

The Fisher kernel is only directly applicable to binary classification tasks. We can apply it to our task by considering an example z to be a sentence-tree pair (x, y), and classifying the pairs into correct parses versus incorrect parses. When we use the Fisher score $\phi_{\hat{\theta}}(x, y)$ in the discriminant function F, we can interpret the value as the confidence that the tree y is correct, and choose the y in which we are the most confident.

2.2 TOP Kernels

Tsuda (2002) proposed another kernel constructed from a probabilistic model, called the Tangent vectors Of Posterior log-odds (TOP) kernel. Their TOP kernel is also only for binary classification tasks, so, as above, we treat the input z as a sentence-tree pair and the output category $c \in \{-1, +1\}$ as incorrect/correct. It is assumed that the true probability distribution is included in the class of probabilistic models and that the true parameter vector $\theta^\star$ is unique. The feature extractor of the TOP kernel for the input z is defined by:

$$\phi_{\hat{\theta}}(z) = (v(z,\hat{\theta}), \frac{\partial v(z,\hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial v(z,\hat{\theta})}{\partial \theta_l}),$$

where $v(z,\hat{\theta}) = \log P(c{=}{+}1|z,\hat{\theta}) - \log P(c{=}{-}1|z,\hat{\theta})$.

In addition to being at least as good as the MAP classifier, the choice of the TOP kernel feature extractor is motivated by the minimization of the binary classification error of a linear classifier $<w, \phi_{\hat{\theta}}(z)> +\ b$. Tsuda (2002) demonstrates that this error is closely related to the estimation error of the posterior probability $P(c{=}{+}1|z,\theta^\star)$ by the estimator $g(<w, \phi_{\hat{\theta}}(z)> +\ b)$, where g is the sigmoid function $g(t) = 1/(1 + \exp(-t))$.

The TOP kernel isn't quite appropriate for structured classification tasks because $\phi_{\hat{\theta}}(z)$ is motivated by binary classificaton error minimization. In the next subsection, we will adapt it to structured classification.

2.3 A TOP Kernel for Reranking

We define the reranking task as selecting a parse tree from the list of candidate trees suggested by a probabilistic model. Furthermore, we only consider learning to rerank the output of a particular probabilistic model, without requiring the classifier to have good performance when applied to a candidate list provided by a different model. In this case, it is natural to model the probability that a parse tree is the best candidate given the list of candidate trees:

$$P(y_k|x, y_1, \ldots, y_s) = \frac{P(x,y_k)}{\sum_t P(x,y_t)},$$

where $y_1, \ldots, y_s$ is the list of candidate parse trees.

To construct a new TOP kernel for reranking, we apply an approach similar to that used for the TOP kernel (Tsuda et al., 2002), but we consider the probability $P(y_k|x, y_1, \ldots, y_s, \theta^\star)$ instead of the probability $P(c{=}{+}1|z,\theta^\star)$ considered by Tsuda. The resulting feature extractor is given by:

$$\phi_{\hat{\theta}}(x, y_k) = (v(x, y_k, \hat{\theta}), \frac{\partial v(x,y_k,\hat{\theta})}{\partial \theta_1}, \ldots, \frac{\partial v(x,y_k,\hat{\theta})}{\partial \theta_l}),$$

where $v(x, y_k, \hat{\theta}) = \log P(y_k|y_1, \ldots, y_s, \hat{\theta}) - \log \sum_{t \neq k} P(y_t|y_1, \ldots, y_s, \hat{\theta})$. We will call this kernel the *TOP reranking kernel*.

3 The Probabilistic Model

To complete the definition of the kernel, we need to choose a probabilistic model of parsing. For this we use a statistical parser which has previously been shown to achieve state-of-the-art performance, namely that proposed in (Henderson, 2003). This parser has two levels of parameterization. The first level of parameterization is in terms of a history-based generative probability model, but this level is not appropriate for our purposes because it defines an infinite number of parameters (one for every possible partial parse history). When parsing a given sentence, the bounded set of parameters which are relevant to a given parse are estimated using a neural network. The weights of this neural network form the second level of parameterization. There is a finite number of these parameters. Neural network training is applied to determine the values of these parameters, which in turn determine the values of the probability model's parameters, which in turn determine the probabilistic model of parse trees.

We do not use the complete set of neural network weights to define our kernels, but instead we define a third level of parameterization which only includes the network's output layer weights. These weights define a normalized exponential model, with the network's hidden layer as the input features. When we tried using the complete set of weights in some small scale experiments, training the classifier was more computationally expensive, and actually performed slightly worse than just using the output weights. Using just the output weights also allows us to make some approximations in the TOP reranking kernel which makes the classifier learning algorithm more efficient.

3.1 A History-Based Probability Model

As with many other statistical parsers (Ratnaparkhi, 1999; Collins, 1999; Charniak, 2000), Henderson (2003) uses a history-based model of parsing. He defines the mapping from phrase structure trees to parse sequences using a form of left-corner parsing strategy (see (Henderson, 2003) for more details). The parser actions include: introducing a new constituent with a specified label, attaching one constituent to another, and predicting the next word of the sentence. A complete parse consists of a sequence of these actions, $d_1,..., d_m$, such that performing $d_1,..., d_m$ results in a complete phrase structure tree.

Because this mapping to parse sequences is one-to-one, and the word prediction actions in a complete parse $d_1,..., d_m$ specify the sentence, $P(d_1,..., d_m)$ is equivalent to the joint probability of the output phrase structure tree and the input sentence. This probability can be then be decomposed into the multiplication of the probabilities of each action decision d_i conditioned on that decision's prior parse history $d_1,..., d_{i-1}$.

$$P(d_1,..., d_m) = \Pi_i P(d_i|d_1,..., d_{i-1})$$

3.2 Estimating Decision Probabilities with a Neural Network

The parameters of the above probability model are the $P(d_i|d_1,..., d_{i-1})$. There are an infinite number of these parameters, since the parse history $d_1,..., d_{i-1}$ grows with the length of the sentence. In other work on history-based parsing, independence assumptions are applied so that only a finite amount of information from the parse history can be treated as relevant to each parameter, thereby reducing the number of parameters to a finite set which can be estimated directly. Instead, Henderson (2003) uses a neural network to induce a finite representation of this unbounded history, which we will denote $h(d_1,..., d_{i-1})$. Neural network training tries to find such a history representation which preserves all the information about the history which is relevant to estimating the desired probability.

$$P(d_i|d_1,..., d_{i-1}) \approx P(d_i|h(d_1,..., d_{i-1}))$$

Using a neural network architecture called Simple Synchrony Networks (SSNs), the history representation $h(d_1,..., d_{i-1})$ is incrementally computed from features of the previous decision d_{i-1} plus a finite set of previous history representations $h(d_1,..., d_j)$, $j < i - 1$. Each history representation is a finite vector of real numbers, called the network's hidden layer. As long as the history representation for position $i - 1$ is always included in the inputs to the history representation for position i, any information about the entire sequence could be passed from history representation to history representation and be used to estimate the desired probability. However, learning is biased towards paying more attention to information which passes through fewer history representations.

To exploit this learning bias, structural locality is used to determine which history representations are input to which others. First, each history representation is assigned to the constituent which is on the top of the parser's stack when it is computed. Then earlier history representations whose constituents are structurally local to the current representation's constituent are input to the computation of the correct representation. In this way, the number of representations which information needs to pass through in order to flow from history representation i to history representation j is determined by the structural distance between i's constituent and j's constituent, and not just the distance between i and j in the parse sequence. This provides the neural network with a linguistically appropriate inductive bias when it learns the history representations, as explained in more detail in (Henderson, 2003).

Once it has computed $h(d_1,..., d_{i-1})$, the SSN uses a normalized exponential to estimate a probability distribution over the set of possible next decisions d_i given the history:

$$P(d_i|d_1,..., d_{i-1}, \theta) \approx \frac{exp(<\theta_{d_i}, h(d_1,.., d_{i-1})>)}{\sum_{t \in N(d_{i-1})} exp(<\theta_t, h(d_1,.., d_{i-1})>)},$$

where by θ_t we denote the set of output layer weights, corresponding to the parser action t, $N(d_{i-1})$ defines a set of possible next parser actions after the step d_{i-1} and θ denotes the full set of model parameters.

We trained SSN parsing models, using the on-line version of Backpropagation to perform the gradient descent with a maximum likelihood objective function. This learning simultaneously tries to optimize the parameters of the output computation and the parameters of the mappings $h(d_1,..., d_{i-1})$. With multi-layered networks such as SSNs, this training is not guaranteed to converge to a global optimum, but in practice a network whose criteria value is close to the optimum can be found.

4 Large-Margin Optimization

Once we have defined a kernel over parse trees, general techniques for linear classifier optimization can be used to learn the given task. The most sophisticated of these techniques (such as Support Vector Machines) are unfortunately too computationally expensive to be used on large datasets like the Penn Treebank (Marcus et al., 1993). Instead we use a method which has often been shown to be virtually as good, the Voted Perceptron (VP) (Freund and Schapire, 1998) algorithm. The VP algorithm was originally applied to parse reranking in (Collins and Duffy, 2002) with the Tree kernel. We modify the perceptron training algorithm to make it more suitable for parsing, where zero-one classification loss is not the evaluation measure usually employed. We also develop a variant of the kernel defined in section 2.3, which is more efficient when used with the VP algorithm.

Given a list of candidate trees, we train the classifier to select the tree with largest constituent F_1 score. The F_1 score is a measure of the similarity between the tree in question and the gold standard parse, and is the standard way to evaluate the accuracy of a parser. We denote the *k*'th candidate tree for the *j*'th sentence x^j by y^j_k. Without loss of generality, let us assume that y^j_1 is the candidate tree with the largest F_1 score.

The Voted Perceptron algorithm is an ensemble method for combining the various intermediate models which are produced during training a perceptron. It demonstrates more stable generalization performance than the normal perceptron algorithm when the problem is not linearly separable (Freund and Schapire, 1998), as is usually the case.

We modify the perceptron algorithm by introducing a new classification loss function. This modification enables us to treat differently the cases where the perceptron predicts a tree with an F_1 score much smaller than that of the top candidate and the cases where the predicted and the top candidates have similar score values. The natural choice for the loss function would be $\Delta(y^j_k, y^j_1) = F_1(y^j_1) - F_1(y^j_k)$, where $F_1(y^j_k)$ denotes the F_1 score value for the parse tree y^j_k. This approach is very similar to slack variable rescaling for Support Vector Machines proposed in (Tsochantaridis et al., 2004). The learning algorithm we employed is presented in figure 1.

When applying kernels with a large training corpus, we face efficiency issues because of the large number of the neural network weights. Even though we use only the output layer weights, this vector grows with the size of the vocabulary, and thus can be large. The kernels presented in section 2 all lead to feature vectors without many zero values. This

```
$w$ = 0
**for** j = 1 .. n
  **for**  k = 2 .. s
    **if** $<w, \phi(x^j, y^j_k)> \; > \; <w, \phi(x^j, y^j_1)>$
      $w$ = $w$ + $\Delta(y^j_k, y^j_1)(\phi(x^j, y^j_1) - \phi(x^j, y^j_k))$
```

Figure 1: The modified perceptron algorithm

happens because we compute the derivative of the normalization factor used in the network's estimation of $P(d_i|d_1,..., d_{i-1})$. This normalization factor depends on the output layer weights corresponding to all the possible next decisions (see section 3.2). This makes an application of the VP algorithm infeasible in the case of a large vocabulary.

We can address this problem by freezing the normalization factor when computing the feature vector. Note that we can rewrite the model log-probability of the tree as:

$$\log P(y|\theta) = \sum_i \log \left(\frac{exp(<\theta_{d_i}, h(d_1,..,d_{i-1})>)}{\sum_{t \in N(d_{i-1})} exp(<\theta_t, h(d_1,..,d_{i-1})>)}\right) = \sum_i (<\theta_{d_i}, h(d_1,..., d_{i-1})>) - \sum_i \log \sum_{t \in N(d_{i-1})} exp(<\theta_t, h(d_1,..., d_{i-1})>).$$

We treat the parameters used to compute the first term as different from the parameters used to compute the second term, and we define our kernel only using the parameters in the first term. This means that the second term does not effect the derivatives in the formula for the feature vector $\phi(x, y)$. Thus the feature vector for the kernel will contain non-zero entries only in the components corresponding to the parser actions which are present in the candidate derivation for the sentence, and thus in the first vector component. We have applied this technique to the TOP reranking kernel, the result of which we will call the *efficient TOP reranking kernel*.

5 The Experimental Results

We used the Penn Treebank WSJ corpus (Marcus et al., 1993) to perform empirical experiments on the proposed parsing models. In each case the input to the network is a sequence of tag-word pairs.[2] We report results for two different vocabulary sizes, varying in the frequency with which tag-word pairs must

[2] We used a publicly available tagger (Ratnaparkhi, 1996) to provide the tags.

occur in the training set in order to be included explicitly in the vocabulary. A frequency threshold of 200 resulted in a vocabulary of 508 tag-word pairs (including tag-unknown_word pairs) and a threshold of 20 resulted in 4215 tag-word pairs. We denote the probabilistic model trained with the vocabulary of 508 by the SSN-Freq≥200, the model trained with the vocabulary of 4215 by the SSN-Freq≥20.

Testing the probabilistic parser requires using a beam search through the space of possible parses. We used a form of beam search which prunes the search after the prediction of each word. We set the width of this post-word beam to 40 for both testing of the probabilistic model and generating the candidate list for reranking. For training and testing of the kernel models, we provided a candidate list consisting of the top 20 parses found by the generative probabilistic model. When using the Fisher kernel, we added the log-probability of the tree given by the probabilistic model as the feature. This was not necessary for the TOP kernels because they already contain a feature corresponding to the probability estimated by the probabilistic model (see section 2.3).

We trained the VP model with all three kernels using the 508 word vocabulary (Fisher-Freq≥200, TOP-Freq≥200, TOP-Eff-Freq≥200) but only the efficient TOP reranking kernel model was trained with the vocabulary of 4215 words (TOP-Eff-Freq≥20). The non-sparsity of the feature vectors for other kernels led to the excessive memory requirements and larger testing time. In each case, the VP model was run for only one epoch. We would expect some improvement if running it for more epochs, as has been empirically demonstrated in other domains (Freund and Schapire, 1998).

To avoid repeated testing on the standard testing set, we first compare the different models with their performance on the validation set. Note that the validation set wasn't used during learning of the kernel models or for adjustment of any parameters.

Standard measures of accuracy are shown in table 1.[3] Both the Fisher kernel and the TOP kernels show better accuracy than the baseline probabilistic

	LR	LP	$F_{\beta=1}$
SSN-Freq≥200	87.2	88.5	87.8
Fisher-Freq≥200	87.2	88.8	87.9
TOP-Freq≥200	87.3	88.9	88.1
TOP-Eff-Freq≥200	87.3	88.9	88.1
SSN-Freq≥20	88.1	89.2	88.6
TOP-Eff-Freq≥20	88.2	89.7	88.9

Table 1: Percentage labeled constituent recall (LR), precision (LP), and a combination of both ($F_{\beta=1}$) on validation set sentences of length at most 100.

model, but only the improvement of the TOP kernels is statistically significant.[4] For the TOP kernel, the improvement over baseline is about the same with both vocabulary sizes. Also note that the performance of the efficient TOP reranking kernel is the same as that of the original TOP reranking kernel, for the smaller vocabulary.

For comparison to previous results, table 2 lists the results on the testing set for our best model (TOP-Efficient-Freq≥20) and several other statistical parsers (Collins, 1999; Collins and Duffy, 2002; Collins and Roark, 2004; Henderson, 2003; Charniak, 2000; Collins, 2000; Shen and Joshi, 2004; Shen et al., 2003; Henderson, 2004; Bod, 2003). First note that the parser based on the TOP efficient kernel has better accuracy than (Henderson, 2003), which used the same parsing method as our baseline model, although the trained network parameters were not the same. When compared to other kernel methods, our approach performs better than those based on the Tree kernel (Collins and Duffy, 2002; Collins and Roark, 2004), and is only 0.2% worse than the best results achieved by a kernel method for parsing (Shen et al., 2003; Shen and Joshi, 2004).

6 Related Work

The first application of kernel methods to parsing was proposed by Collins and Duffy (2002). They used the Tree kernel, where the features of a tree are all its connected tree fragments. The VP algorithm was applied to rerank the output of a probabilistic model and demonstrated an improvement over the baseline.

[3]All our results are computed with the evalb program following the standard criteria in (Collins, 1999), and using the standard training (sections 2–22, 39,832 sentences, 910,196 words), validation (section 24, 1346 sentence, 31507 words), and testing (section 23, 2416 sentences, 54268 words) sets (Collins, 1999).

[4]We measured significance with the randomized significance test of (Yeh, 2000).

	LR	LP	F$_{\beta=1^*}$
Collins99	88.1	88.3	88.2
Collins&Duffy02	88.6	88.9	88.7
Collins&Roark04	88.4	89.1	88.8
Henderson03	88.8	89.5	89.1
Charniak00	89.6	89.5	89.5
TOP-Eff-Freq$\geq$20	89.1	90.1	89.6
Collins00	89.6	89.9	89.7
Shen&Joshi04	89.5	90.0	89.8
Shen et al.03	89.7	90.0	89.8
Henderson04	89.8	90.4	90.1
Bod03	90.7	90.8	90.7

* F$_{\beta=1}$ for previous models may have rounding errors.

Table 2: Percentage labeled constituent recall (LR), precision (LP), and a combination of both (F$_{\beta=1}$) on the entire testing set.

Shen and Joshi (2003) applied an SVM based voting algorithm with the Preference kernel defined over pairs for reranking. To define the Preference kernel they used the Tree kernel and the Linear kernel as its underlying kernels and achieved state-of-the-art results with the Linear kernel.

In (Shen et al., 2003) it was pointed out that most of the arbitrary tree fragments allowed by the Tree kernel are linguistically meaningless. The authors suggested the use of Lexical Tree Adjoining Grammar (LTAG) based features as a more linguistically appropriate set of features. They empirically demonstrated that incorporation of these features helps to improve reranking performance.

Shen and Joshi (2004) proposed to improve margin based methods for reranking by defining the margin not only between the top tree and all the other trees in the candidate list but between all the pairs of parses in the ordered candidate list for the given sentence. They achieved the best results when training with an uneven margin scaled by the heuristic function of the candidates positions in the list. One potential drawback of this method is that it doesn't take into account the actual F_1 score of the candidate and considers only the position in the list ordered by the F_1 score. We expect that an improvement could be achieved by combining our approach of scaling updates by the F_1 loss with the all pairs approach of (Shen and Joshi, 2004). Use of the F_1 loss function during training demonstrated better performance comparing to the 0-1 loss function when applied to a structured classification task (Tsochantaridis et al., 2004).

All the described kernel methods are limited to the reranking of candidates from an existing parser due to the complexity of finding the best parse given a kernel (i.e. the decoding problem). (Taskar et al., 2004) suggested a method for maximal margin parsing which employs the dynamic programming approach to decoding and parameter estimation problems. The efficiency of dynamic programming means that the entire space of parses can be considered, not just a candidate list. However, not all kernels are suitable for this method. The dynamic programming approach requires the feature vector of a tree to be decomposable into a sum over parts of the tree. In particular, this is impossible with the TOP and Fisher kernels derived from the SSN model. Also, it isn't clear whether the algorithm remains tractable for a large training set with long sentences, since the authors only present results for sentences of length less than or equal to 15.

7 Conclusions

This paper proposes a method for deriving a kernel for reranking from a probabilistic model, and demonstrates state-of-the-art accuracy when this method is applied to parse reranking. Contrary to most of the previous research on kernel methods in parsing, linguistic knowledge does not have to be expressed through a list of features, but instead can be expressed through the design of a probability model. The parameters of this probability model are then trained, so that they reflect what features of trees are relevant to parsing. The kernel is then derived from this trained model in such a way as to maximize its usefulness for reranking.

We performed experiments on parse reranking using a neural network based statistical parser as both the probabilistic model and the source of the list of candidate parses. We used a modification of the Voted Perceptron algorithm to perform reranking with the kernel. The results were amongst the best current statistical parsers, and only 0.2% worse than the best current parsing methods which use kernels. We would expect further improvement if we used different models to derive the kernel and to generate the candidates, thereby exploiting the advantages of combining multiple models, as do the better performing methods using kernels.

In recent years, probabilistic models have become commonplace in natural language processing. We believe that this approach to defining kernels would simplify the problem of defining kernels for these tasks, and could be very useful for many of them. In particular, maximum entropy models also use a normalized exponential function to estimate probabilities, so all the methods discussed in this paper would be applicable to maximum entropy models. This approach would be particularly useful for tasks where there is less data available than in parsing, for which large-margin methods work particularly well.

References

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proc. 10th Conf. of European Chapter of the Association for Computational Linguistics*, Budapest, Hungary.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. 1st Meeting of North American Chapter of Association for Computational Linguistics*, pages 132–139, Seattle, Washington.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures and the voted perceptron. In *Proc. 40th Meeting of Association for Computational Linguistics*, pages 263–270.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. 42th Meeting of Association for Computational Linguistics*, Barcelona, Spain.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. 17th Int. Conf. on Machine Learning*, pages 175–182, Stanford, CA.

Yoav Freund and Robert E. Schapire. 1998. Large margin classification using the perceptron algorithm. In *Proc. of the 11th Annual Conf. on Computational Learning Theory*, pages 209–217, Madisson WI.

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.

Tommi S. Jaakkola and David Haussler. 1998. Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processes Systems 11*.

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.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. Conf. on Empirical Methods in Natural Language Processing*, pages 133–142, Univ. of Pennsylvania, PA.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175.

Libin Shen and Aravind K. Joshi. 2003. An SVM based voting algorithm with application to parse reranking. In *Proc. of the 7th Conf. on Computational Natural Language Learning*, pages 9–16, Edmonton, Canada.

Libin Shen and Aravind K. Joshi. 2004. Flexible margin selection for reranking with full pairwise samples. In *Proc. of the 1st Int. Joint Conf. on Natural Language Processing*, Hainan Island, China.

Libin Shen, Anoop Sarkar, and Aravind K. Joshi. 2003. Using LTAG based features in parse reranking. In *Proc. of Conf. on Empirical Methods in Natural Language Processing*, Sapporo, Japan.

Ben Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In *Proc. Conf. on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. 21st Int. Conf. on Machine Learning*, pages 823–830, Banff, Alberta, Canada.

K. Tsuda, M. Kawanabe, G. Ratsch, S. Sonnenburg, and K. Muller. 2002. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10):2397–2414.

Alexander Yeh. 2000. More accurate tests for the statistical significance of the result differences. In *Proc. 17th International Conf. on Computational Linguistics*, pages 947–953, Saarbruken, Germany.

Boosting-based parse reranking with subtree features

Taku Kudo [*] **Jun Suzuki** **Hideki Isozaki**
NTT Communication Science Laboratories.
2-4 Hikaridai, Seika-cho, Soraku, Kyoto, Japan
{taku,jun,isozaki}@cslab.kecl.ntt.co.jp

Abstract

This paper introduces a new application of boosting for parse reranking. Several parsers have been proposed that utilize the all-subtrees representation (e.g., tree kernel and data oriented parsing). This paper argues that such an all-subtrees representation is extremely redundant and a comparable accuracy can be achieved using just a small set of subtrees. We show how the boosting algorithm can be applied to the all-subtrees representation and how it selects a small and relevant feature set efficiently. Two experiments on parse reranking show that our method achieves comparable or even better performance than kernel methods and also improves the testing efficiency.

1 Introduction

Recent work on statistical natural language parsing and tagging has explored *discriminative* techniques. One of the novel discriminative approaches is *reranking*, where discriminative machine learning algorithms are used to rerank the n-best outputs of generative or conditional parsers. The discriminative reranking methods allow us to incorporate various kinds of features to distinguish the correct parse tree from all other candidates.

With such feature design flexibility, it is non-trivial to employ an appropriate feature set that has a good discriminative ability for parse reranking. In early studies, feature sets were given heuristically by simply preparing task-dependent *feature templates* (Collins, 2000; Collins, 2002). These ad-hoc solutions might provide us with reasonable levels of performance. However, they are highly task dependent and require careful design to create the optimal feature set for each task. Kernel methods offer an elegant solution to these problems. They can work on a potentially huge or even infinite number of features without a loss of generalization. The best known kernel for modeling a tree is the tree kernel (Collins and Duffy, 2002), which argues that a feature vector is implicitly composed of the counts of subtrees. Although kernel methods are general and can cover almost all useful features, the set of subtrees that is used is extremely redundant. The main question addressed in this paper concerns whether it is possible to achieve a comparable or even better accuracy using just a small and non-redundant set of subtrees.

In this paper, we present a new application of boosting for parse reranking. While tree kernel *implicitly* uses the all-subtrees representation, our boosting algorithm uses it *explicitly*. Although this set-up makes the feature space large, the l_1-norm regularization achived by boosting automatically selects a small and relevant feature set. Such a small feature set is useful in practice, as it is interpretable and makes the parsing (reranking) time faster. We also incorporate a variant of the branch-and-bound technique to achieve efficient feature selection in each boosting iteration.

2 General setting of parse reranking

We describe the general setting of parse reranking.

- Training data T is a set of input/output pairs, e.g., $T = \{\langle \mathbf{x}_1, \mathbf{y}_1 \rangle, \ldots, \langle \mathbf{x}_L, \mathbf{y}_L \rangle\}$, where $\mathbf{x}_i$ is an input sentence, and $\mathbf{y}_i$ is a correct parse associated with the sentence $\mathbf{x}_i$.

- Let $\mathcal{Y}(\mathbf{x})$ be a function that returns a set of candi-

[*]Currently, Google Japan Inc., taku@google.com

date parse trees for a particular sentence $\mathbf{x}$.

- We assume that $\mathcal{Y}(\mathbf{x}_i)$ contains the correct parse tree $\mathbf{y}_i$, i.e., $\mathbf{y}_i \in \mathcal{Y}(\mathbf{x}_i)$ [*]
- Let $\Phi(\mathbf{y}) \in \mathbb{R}^d$ be a feature function that maps the given parse tree $\mathbf{y}$ into $\mathbb{R}^d$ space. $\mathbf{w} \in \mathbb{R}^d$ is a parameter vector of the model. The output parse $\hat{\mathbf{y}}$ of this model on input sentence $\mathbf{x}$ is given as: $\hat{\mathbf{y}} = \mathrm{argmax}_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \mathbf{w} \cdot \Phi(\mathbf{y})$.

There are two questions as regards this formulation. One is how to set the parameters $\mathbf{w}$, and the other is how to design the feature function $\Phi(\mathbf{y})$. We briefly describe the well-known solutions to these two problems in the next subsections.

2.1 Parameter estimation

We usually adopt a general loss function $Loss(\mathbf{w})$, and set the parameters $\mathbf{w}$ that minimize the loss, i.e., $\hat{\mathbf{w}} = \mathrm{argmin}_{\mathbf{w}\in\mathbb{R}^d} Loss(\mathbf{w})$. Generally, the loss function has the following form:

$$Loss(\mathbf{w}) = \sum_{i=1}^{L} L(\mathbf{w}, \Phi(\mathbf{y}_i), \mathbf{x}_i),$$

where $L(\mathbf{w}, \Phi(\mathbf{y}_i), \mathbf{x}_i)$ is an arbitrary loss function. We can design a variety of parameter estimation methods by changing the loss function. The following three loss functions, $LogLoss$, $HingeLoss$, and $BoostLoss$, have been widely used in parse reranking tasks.

$$\begin{aligned}
LogLoss &= -\log \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}_i)} \exp \quad \mathbf{w} \cdot [\Phi(\mathbf{y}_i) - \Phi(\mathbf{y})] \\
HingeLoss &= \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}_i)} \max(0, 1 - \mathbf{w} \cdot [\Phi(\mathbf{y}_i) - \Phi(\mathbf{y})]) \\
BoostLos &= \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}_i)} \exp \quad -\mathbf{w} \cdot [\Phi(\mathbf{y}_i) - \Phi(\mathbf{y})]
\end{aligned}$$

$LogLoss$ is based on the standard maximum likelihood optimization, and is used with maximum entropy models. $HingeLoss$ captures the errors only when $\mathbf{w} \cdot [\Phi(\mathbf{y}_i) - \Phi(\mathbf{y})]) < 1$. This loss is closely related to the maximum margin strategy in SVMs (Vapnik, 1998). $BoostLoss$ is analogous to the boosting algorithm and is used in (Collins, 2000; Collins, 2002).

[*]In the real setting, we cannot assume this condition. In this case, we select the parse tree $\hat{\mathbf{y}}$ that is the most similar to $\mathbf{y}_i$ and take $\hat{\mathbf{y}}$ as the correct parse tree $\mathbf{y}_i$.

2.2 Definition of feature function

It is non-trivial to define an appropriate feature function $\Phi(\mathbf{y})$ that has a good ability to distinguish the correct parse $\mathbf{y}_i$ from all other candidates

In early studies, the feature functions were given heuristically by simply preparing *feature templates* (Collins, 2000; Collins, 2002). However, such heuristic selections are task dependent and would not cover all useful features that contribute to overall accuracy.

When we select the special family of loss functions, the problem can be reduced to a dual form that depends only on the inner products of two instances $\Phi(\mathbf{y}_1) \cdot \Phi(\mathbf{y}_2)$. This property is important as we can use a *kernel trick* and we do not need to provide an explicit feature function. For example, tree kernel (Collins and Duffy, 2002), one of the convolution kernels, implicitly maps the instance represented in a tree into all-subtrees space. Even though the feature space is large, inner products under this feature space can be calculated efficiently using dynamic programming. Tree kernel is more general than feature templates since it can use the all-subtrees representation without loss of efficiency.

3 RankBoost with subtree features

A simple question related to kernel-based parse reranking asks whether *all* subtrees are really needed to construct the final parameters $\mathbf{w}$. Suppose we have two *large* trees t and t', where t' is simply generated by attaching a single node to t. In most cases, these two trees yield an almost equivalent discriminative ability, since they are very similar and highly correlated with each other. Even when we exploit all subtrees, most of them are extremely redundant.

The motivation of this paper is based on the above observation. We think that only a small set of subtrees is needed to express the final parameters. A compact, non-redundant, and highly relevant feature set is useful in practice, as it is interpretable and increases the parsing (reranking) speed.

To realize this goal, we propose a new boosting-based reranking algorithm based on the all-subtrees representation. First, we describe the architecture of our reranking method. Second, we show a connection between boosting and SVMs, and describe how the algorithm realizes the sparse feature representa-

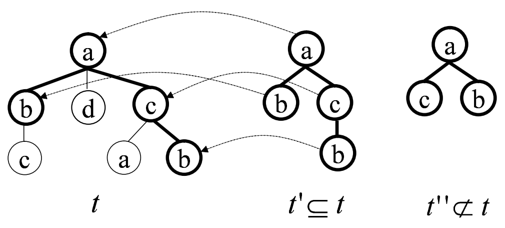


Figure 1: Labeled ordered tree and subtree relation

tion described above.

3.1 Preliminaries

Let us introduce a labeled ordered tree (or simply ’tree’), its definition and notations, first.

Definition 1 *Labeled ordered tree (Tree)*
A labeled ordered tree is a tree where each node is associated with a label and is ordered among its siblings, that is, there is a first child, second child, third child, etc.

Definition 2 *Subtree*
Let t and u be labeled ordered trees. We say that t matches u, or t is a subtree of u ($t \subseteq u$), if there is a one-to-one function ψ from nodes in t to u, satisfying the conditions: (1) ψ preserves the parent-daughter relation, (2) ψ preserves the sibling relation, (3) ψ preserves the labels.

We denote the number of nodes in t as $|t|$. Figure 1 shows an example of a labeled ordered tree and its subtree and non-subtree.

3.2 Feature space given by subtrees

We first assume that a parse tree $\mathbf{y}$ is represented in a labeled ordered tree. Note that the outputs of part-of-speech tagging, shallow parsing, and dependency analysis can be modeled as labeled ordered trees.

The feature set $\mathcal{F}$ consists of all subtrees seen in the training data, i.e.,

$$\mathcal{F} = \cup_{i, \mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \{ t \mid t \subseteq \mathbf{y} \}.$$

The feature mapping $\Phi(\mathbf{y})$ is then given by letting the existence of a tree t be a single dimension, i.e.,

$$\Phi(\mathbf{y}) = \{ I(t_1 \subseteq \mathbf{y}), \ldots, I(t_m \subseteq \mathbf{y}) \} \in \{0,1\}^m,$$

where $I(\cdot)$ is the indicator function, $m = |\mathcal{F}|$, and $\{t_1, \ldots, t_m\} \in \mathcal{F}$. The feature space is essentially the same as that of tree kernel [†]

[†]Strictly speaking, tree kernel uses the cardinality of each subtree

3.3 RankBoost algorithm

The parameter estimation method we adopt is a variant of the RankBoost algorithm introduced in (Freund et al., 2003). Collins et al. used RankBoost to parse reranking tasks (Collins, 2000; Collins, 2002). The algorithm proceeds for K iterations and tries to minimize the $BoostLoss$ for given training data[‡]. At each iteration, a single feature (hypothesis) is chosen, and its weight is updated.

Suppose we have current parameters:

$$\mathbf{w} = \{ w_1, w_2, \ldots, w_m \} \in \mathbb{R}^m.$$

New parameters $\mathbf{w}^*_{\langle k, \delta \rangle} \in \mathbb{R}^m$ are then given by selecting a single feature k and updating the weight through an increment δ:

$$\mathbf{w}^*_{\langle k, \delta \rangle} = \{ w_1, w_2, \ldots, w_k + \delta, \ldots, w_m \}.$$

After the update, the new loss is given:

$$Loss(\mathbf{w}^*_{\langle k, \delta \rangle}) = \sum_{i,\ \mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} \exp \left(-\mathbf{w}^*_{\langle k, \delta \rangle} \cdot [\Phi(\mathbf{y}_i) - \Phi(\mathbf{y})] \right). \quad (1)$$

The RankBoost algorithm iteratively selects the optimal pair $\langle \hat{k}, \hat{\delta} \rangle$ that minimizes the loss, i.e.,

$$\langle \hat{k}, \hat{\delta} \rangle = \operatorname*{argmin}_{\langle k, \delta \rangle} Loss(\mathbf{w}^*_{\langle k, \delta \rangle}).$$

By setting the differential of (1) at 0, the following optimal solutions are obtained:

$$\hat{k} = \operatorname*{argmax}_{k=1,\ldots,m} \left| \sqrt{W_k^+} - \sqrt{W_k^-} \right|, \text{ and } \delta = \frac{1}{2} \log \frac{W_{\hat{k}}^+}{W_{\hat{k}}^-}, \quad (2)$$

where $W_k^b = \sum_{i, \mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} D(\mathbf{y}_i, \mathbf{y}) \cdot I[I(t_k \subseteq \mathbf{y}_i) - I(t_k \subseteq \mathbf{y}) = b]$, $b \in \{+1, -1\}$, and $D(\mathbf{y}_i, \mathbf{y}) = \exp\left(-\mathbf{w} \cdot [\Phi(\mathbf{y}_i) - \Phi(\mathbf{y})] \right)$.

Following (Freund et al., 2003; Collins, 2000), we introduce smoothing to prevent the case when either W_k^+ or W_k^- is 0 [§]:

$$\delta = \frac{1}{2} \log \frac{W_{\hat{k}}^+ + \epsilon Z}{W_{\hat{k}}^- + \epsilon Z}, \text{ where } Z = \sum_{i, \mathbf{y} \in \mathcal{Y}(\mathbf{x}_i)} D(\mathbf{y}_i, \mathbf{y}) \text{ and } \epsilon \in \mathbb{R}^+.$$

The function $\mathcal{Y}(\mathbf{x})$ is usually performed by a probabilistic history-based parser, which can output not only a parse tree but the log probability of the

[‡]In our experiments, optimal settings for K were selected by using development data.

[§]For simplicity, we fix ϵ at 0.001 in all our experiments.

tree. We incorporate the log probability into the reranking by using it as a feature:

$$\begin{aligned}\Phi(\mathbf{y}) &= \{L(\mathbf{y}), I(t_1 \subseteq \mathbf{y}), \ldots, I(t_m \subseteq \mathbf{y})\}, \text{ and} \\ \mathbf{w} &= \{w_0, w_1, w_2, \ldots, w_m\},\end{aligned}$$

where $L(\mathbf{y})$ is the log probability of a tree $\mathbf{y}$ under the base parser and w_0 is the parameter of $L(\mathbf{y})$. Note that the update algorithm (2) does not allow us to calculate the parameter w_0, since (2) is restricted to binary features. To prevent this problem, we use the approximation technique introduced in (Freund et al., 2003).

3.4 Sparse feature representation

Recent studies (Schapire et al., 1997; Rätsch, 2001) have shown that both boosting and SVMs (Vapnik, 1998) work according to similar strategies: constructing optimal parameters $\mathbf{w}$ that maximize the *smallest margin* between positive and negative examples. The critical difference is the definition of margin or the way they regularize the vector $\mathbf{w}$. (Rätsch, 2001) shows that the iterative feature selection performed in boosting asymptotically realizes an l_1-norm $||\mathbf{w}||_1$ regularization. In contrast, it is well known that SVMs are reformulated as an l_2-norm $||\mathbf{w}||_2$ regularized algorithm.

The relationship between two regularizations has been studied in the machine learning community. (Perkins et al., 2003) reported that l_1-norm should be chosen for a problem where most given features are *irrelevant*. On the other hand, l_2-norm should be chosen when most given features are *relevant*. An advantage of the l_1-norm regularizer is that it often leads to sparse solutions where most w_k are exactly 0. The features assigned zero weight are thought to be *irrelevant* features as regards classifications.

The l_1-norm regularization is useful for our setting, since most features (subtrees) are redundant and irrelevant, and these redundant features are automatically eliminated.

4 Efficient Computation

In each boosting iteration, we have to solve the following optimization problem:

$$\hat{k} = \underset{k=1,\ldots,m}{\text{argmax}}\; gain(t_k),$$

$$\text{where } gain(t_k) = \left|\sqrt{W_k^+} - \sqrt{W_k^-}\right|.$$

It is non-trivial to find the optimal tree $t_{\hat{k}}$ that maximizes $gain(t_k)$, since the number of subtrees is exponential to its size. In fact, the problem is known to be NP-hard (Yang, 2004). However, in real applications, the problem is manageable, since the maximum number of subtrees is usually bounded by a constant. To solve the problem efficiently, we now adopt a variant of the branch-and-bound algorithm, similar to that described in (Kudo and Matsumoto, 2004)

4.1 Efficient Enumeration of Trees

Abe and Zaki independently proposed an efficient method, *rightmost-extension*, for enumerating all subtrees from a given tree (Abe et al., 2002; Zaki, 2002). First, the algorithm starts with a set of trees consisting of single nodes, and then expands a given tree of size $(n-1)$ by attaching a new node to it to obtain trees of size n. However, it would be inefficient to expand nodes at arbitrary positions of the tree, as duplicated enumeration is inevitable. The algorithm, rightmost extension, avoids such duplicated enumerations by restricting the position of attachment. Here we give the definition of rightmost extension to describe this restriction in detail.

Definition 3 *Rightmost Extension (Abe et al., 2002; Zaki, 2002)*
Let t and t' be labeled ordered trees. We say t' is a rightmost extension of t, if and only if t and t' satisfy the following three conditions:
(1) t' is created by adding a single node to t, (i.e., $t \subset t'$ and $|t| + 1 = |t'|$).
(2) A node is added to a node existing on the unique path from the root to the rightmost leaf (rightmost-path) in t.
(3) A node is added as the rightmost sibling.

Consider Figure 2, which illustrates example tree t with labels drawn from the set $\mathcal{L} = \{a, b, c\}$. For the sake of convenience, each node in this figure has its original number (depth-first enumeration). The rightmost-path of the tree t is $(a(c(b)))$, and it occurs at positions $1, 4$ and 6 respectively. The set of rightmost extended trees is then enumerated by simply adding a single node to a node on the rightmost path. Since there are three nodes on the rightmost path and the size of the label set is 3 ($= |\mathcal{L}|$), a to-

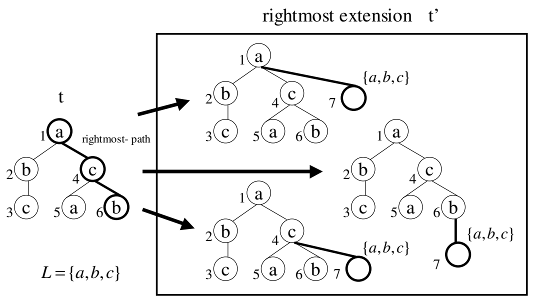


Figure 2: Rightmost extension

tal of 9 trees are enumerated from the original tree t. By repeating the rightmost-extension process recursively, we can create a search space in which all trees drawn from the set $\mathcal{L}$ are enumerated.

4.2 Pruning

Rightmost extension defines a canonical search space in which we can enumerate all subtrees from a given set of trees. Here we consider an upper bound of the gain that allows subspace pruning in this canonical search space. The following observation provides a convenient way of computing an upper bound of the $gain(t_k)$ for any super-tree $t_{k'}$ of t_k.

Observation 1 *Upper bound of the* $gain(t_k)$
For any $t_{k'} \supseteq t_k$*, the gain of* $t_{k'}$ *is bounded by* $\mu(t_k)$*:*

$$\begin{aligned} gain(t_{k'}) &= \sqrt{W^+_{k'}} - \sqrt{W^-_{k'}} \\ &\le \max(\sqrt{W^+_{k'}}, \sqrt{W^-_{k'}}) \\ &\le \max(\sqrt{W^+_{k}}, \sqrt{W^-_{k}}) = \mu(t_k), \\ \textit{since} \quad & t_{k'} \supseteq t_k \Rightarrow W^b_{k'} \le W^b_k,\ b \in \{+1, -1\}. \end{aligned}$$

We can efficiently prune the search space spanned by the rightmost extension using the upper bound of gain $\mu(t)$. During the traverse of the subtree lattice built by the recursive process of rightmost extension, we always maintain the temporally suboptimal gain τ of all the previously calculated gains. If $\mu(t) < \tau$, the gain of any super-tree $t' \supseteq t$ is no greater than τ, and therefore we can safely prune the search space spanned from the subtree t. In contrast, if $\mu(t) \ge \tau$, we cannot prune this space, since there might be a super-tree $t' \supseteq t$ such that $gain(t') \ge \tau$.

4.3 Ad-hoc techniques

In real applications, we also employ the following practical methods to reduce the training costs.

- Size constraint
  Larger trees are usually less effective to discrimination. Thus, we give a size threshold s, and use subtrees whose size is no greater than s. This constraint is easily realized by controlling the rightmost extension according to the size of the trees.
- Frequency constraint
  The frequency-based cut-off has been widely used in feature selections. We employ a frequency threshold f, and use subtrees seen on at least one parse for at least f different sentences. Note that a similar branch-and-bound technique can also be applied to the cut-off. When we find that the frequency of a tree t is no greater than f, we can safely prune the space spanned from t as the frequencies of any super-trees $t' \supseteq t$ are also no greater than f.
- Pseudo iterations
  After several 5- or 10-iterations of boosting, we alternately perform 100- or 300 pseudo iterations, in which the optimal feature (subtree) is selected from the cache that maintains the features explored in the previous iterations. The idea is based on our observation that a feature in the cache tends to be reused as the number of boosting iterations increases. Pseudo iterations converge very fast, and help the branch-and-bound algorithm find new features that are not in the cache.

5 Experiments

5.1 Parsing Wall Street Journal Text

In our experiments, we used the same data set that used in (Collins, 2000). Sections 2-21 of the Penn Treebank were used as training data, and section 23 was used as test data. The training data contains about 40,000 sentences, each of which has an average of 27 distinct parses. Of the 40,000 training sentences, the first 36,000 sentences were used to perform the RankBoost algorithm. The remaining 4,000 sentences were used as development data. Model2 of (Collins, 1999) was used to parse both the training and test data.

To capture the lexical information of the parse trees, we did not use a standard CFG tree but a lexicalized-CFG tree where each non-terminal node has an extra lexical node labeled with the head word of the constituent. Figure 3 shows an example of the lexicalized-CFG tree used in our experiments. The

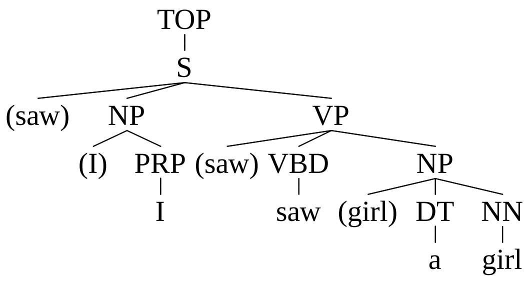


Figure 3: Lexicalized CFG tree for WSJ parsing
head word, e.g., (saw), is put as a leftmost constituent

size parameter s and frequency parameter f were experimentally set at 6 and 10, respectively. As the data set is very large, it is difficult to employ the experiments with more unrestricted parameters.

Table 1 lists results on test data for the Model2 of (Collins, 1999), for several previous studies, and for our best model. We achieve recall and precision of 89.3/%89.6% and 89.9%/90.1% for sentences with $\leq$ 100 words and $\leq$ 40 words, respectively. The method shows a 1.2% absolute improvement in average precision and recall (from 88.2% to 89.4% for sentences $\leq$ 100 words), a 10.1% relative reduction in error. (Collins, 2000) achieved 89.6%/89.9% recall and precision for the same datasets (sentences $\leq$ 100 words) using boosting and manually constructed features. (Charniak, 2000) extends PCFG and achieves similar performance to (Collins, 2000). The tree kernel method of (Collins and Duffy, 2002) uses the all-subtrees representation and achieves 88.6%/88.9% recall and precision, which are slightly worse than the results obtained with our model. (Bod, 2001) also uses the all-subtrees representation with a very different parameter estimation method, and realizes 90.06%/90.08% recall and precision for sentences of ≤ 40 words.

5.2 Shallow Parsing

We used the same data set as the CoNLL 2000 shared task (Tjong Kim Sang and Buchholz, 2000). Sections 15-18 of the Penn Treebank were used as training data, and section 20 was used as test data.

As a baseline model, we used a shallow parser based on Conditional Random Fields (CRFs), very similar to that described in (Sha and Pereira, 2003). CRFs have shown remarkable results in a number of tagging and chunking tasks in NLP. n-best outputs were obtained by a combination of forward Viterbi search and backward A* search. Note that this search algorithm yields optimal n-best results in terms of the CRFs score. Each sentence has at most 20 distinct parses. The log probability from the CRFs shallow parser was incorporated into the reranking. Following (Collins, 2000), the training set was split into 5 portions, and the CRFs shallow parser was trained on 4/5 of the data, then used to decode the remaining 1/5. The outputs of the base parser, which consist of base phrases, were converted into right-branching trees by assuming that two adjacent base phrases are in a parent-child relationship. Figure 4 shows an example of the tree for shallow parsing task. We also put two virtual nodes, left/right boundaries, to capture local transitions. The size parameter s and frequency parameter f were experimentally set at 6 and 5, respectively.

Table 2 lists results on test data for the baseline CRFs parser, for several previous studies, and for our best model. Our model achieves a 94.12 F-measure, and outperforms the baseline CRFs parser and the SVMs parser (Kudo and Matsumoto, 2001). (Zhang et al., 2002) reported a higher F-measure with a generalized winnow using additional linguistic features. The accuracy of our model is very similar to that of (Zhang et al., 2002) without using such additional features. Table 3 shows the results for our best model per chunk type.

MODEL	≤ 40 Words (2245 sentences)				
	LR	LP	CBs	0 CBs	2 CBs
CO99	88.5%	88.7%	0.92	66.7%	87.1%
CH00	90.1%	90.1%	0.74	70.1%	89.6%
CO00	90.1%	90.4%	0.74	70.3%	89.6%
CO02	89.1%	89.4%	0.85	69.3%	88.2%
Boosting	89.9%	90.1%	0.77	70.5%	89.4%

MODEL	≤ 100 Words (2416 sentences)				
	LR	LP	CBs	0 CBs	2 CBs
CO99	88.1%	88.3%	1.06	64.0%	85.1%
CH00	89.6%	89.5%	0.88	67.6%	87.7%
CO00	89.6%	89.9%	0.87	68.3%	87.7%
CO02	88.6%	88.9%	0.99	66.5%	86.3%
Boosting	89.3%	89.6%	0.90	67.9%	87.5%

Table 1: Results for section 23 of the WSJ Treebank
LR/LP = labeled recall/precision. CBs is the average number of cross brackets per sentence. 0 CBs, and 2CBs are the percentage of sentences with 0 or $\leq$ 2 crossing brackets, respectively. COL99 = Model 2 of (Collins, 1999). CH00 = (Charniak, 2000), CO00=(Collins, 2000). CO02=(Collins and Duffy, 2002).

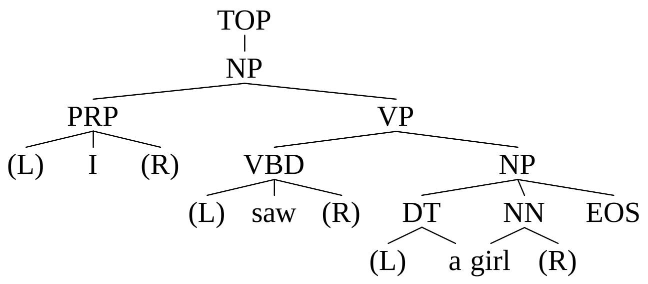


Figure 4: Tree representation for shallow parsing
Represented in a right-branching tree with two virtual nodes

MODEL	$F_{\beta=1}$
CRFs (baseline)	93.76
8 SVMs-voting (Kudo and Matsumoto, 2001)	93.91
RW + linguistic features (Zhang et al., 2002)	94.17
Boosting (our model)	94.12

Table 2: Results of shallow parsing
$F_{\beta=1}$ is the harmonic mean of precision and recall.

6 Discussion

6.1 Interpretablity and Efficiency

The numbers of active (non-zero) features selected by boosting are around 8,000 and 3,000 in the WSJ parsing and shallow parsing, respectively. Although almost all the subtrees are used as feature candidates, boosting selects a small and highly relevant subset of features. When we explicitly enumerate the subtrees used in tree kernel, the number of active features might amount to millions or more. Note that the accuracies under such sparse feature spaces are still comparable to those obtained with tree kernel. This result supports our first intuition that we do not always need all the subtrees to construct the parameters.

The sparse feature representations are useful in practice as they allow us to analyze what kinds of features are relevant. Table 4 shows examples of active features along with their weights w_k. In the shallow parsing tasks, subordinate phrases (SBAR) are difficult to analyze without seeing long dependencies. Subordinate phrases usually precede a sentence (NP and VP). However, Markov-based shallow parsers, such as MEMM or CRFs, cannot capture such a long dependency. Our model automatically selects useful subtrees to obtain an improvement on subordinate phrases. It is interesting that the

	Precision	Recall	$F_{\beta=1}$
ADJP	80.35%	73.41%	76.72
ADVP	83.88%	82.33%	83.10
CONJP	42.86%	66.67%	52.17
INTJ	50.00%	50.00%	50.00
LST	0.00%	0.00%	0.00
NP	94.45%	94.36%	94.41
PP	97.24%	98.07%	97.65
PRT	76.92%	75.47%	76.19
SBAR	90.70%	89.35%	90.02
VP	93.95%	94.72%	94.33
Overall	94.11%	94.13%	94.12

Table 3: Results of shallow parsing per chunk type

tree (SBAR(IN(for))(NP(VP(TO)))) has a large positive weight, while the tree (SBAR((IN(for))(NP(O)))) has a negative weight. The improvement on subordinate phrases is considerable. We achieve 19% of the relative error reduction for subordinate phrase (from 87.68 to 90.02 in F-measure)

The testing speed of our model is much higher than that of other models. The speeds of reranking for WSJ parsing and shallow parsing are 0.055 sec./sent. and 0.042 sec./sent. respectively, which are fast enough for real applications ¶.

6.2 Relationship to previous work

Tree kernel uses the all-subtrees representation not explicitly but implicitly by reducing the problem to the calculation of the inner-products of two trees. The implicit calculation yields a practical computation in training. However, in testing, kernel methods require a number of kernel evaluations, which are too heavy to allow us to realize real applications. Moreover, tree kernel needs to incorporate a decay factor to downweight the contribution of larger subtrees. It is non-trivial to set the optimal decay factor as the accuracies are sensitive to its selection.

Similar to our model, data oriented parsing (DOP) methods (Bod, 1998) deal with the all-subtrees representation explicitly. Since the exact computation of scores for DOP is NP-complete, several approximations are employed to perform an efficient parsing. The critical difference between our model and DOP is that our model leads to an extremely sparse solution and automatically eliminates redundant subtrees. With the DOP methods, (Bod, 2001) also employs constraints (e.g., depth of subtrees) to

¶We ran these tests on a Linux PC with Pentium 4 3.2 Ghz.

WSJ parsing	
w	active trees that contain the word *"in"*
0.3864	(VP(NP(NNS(plants)))(PP(in)))
0.3326	(VP(VP(PP)(PP(in)))(VP))
0.2196	(NP(VP(VP(PP)(PP(in)))))
0.1748	(S(NP(NNP))(PP(in)(NP)))
...	...
-1.1217	(PP(in)(NP(NP(effect))))
-1.1634	(VP(yield)(PP(PP))(PP(in)))
-1.3574	(NP(PP(in)(NP(NN(way)))))
-1.8030	(NP(PP(in)(NP(trading)(JJ))))

shallow parsing	
w	active trees that contain the phrase *"SBAR"*
1.4500	(SBAR(IN(for))(NP(VP(TO))))
0.6177	(VP(SBAR(NP(VBD)))
0.6173	(SBAR(NP(VP(“))))
0.5644	(VP(SBAR(NP(VP(JJ)))))
..	..
-0.9034	(SBAR(IN(for))(NP(O)))
-0.9181	(SBAR(NP(O)))
-1.0695	(ADVP(NP(SBAR(NP(VP)))))
-1.1699	(SBAR(NP(NN)(NP)))

Table 4: Examples of active features (subtrees)

All trees are represented in S-expression. In the shallow parsing task, **O** is a special phrase that means “out of chunk”.

select relevant subtrees and achieves the best results for WSJ parsing. However, these techniques are not based on the regularization framework focused on this paper and do not always eliminate all the redundant subtrees. Even using the methods of (Bod, 2001), millions of subtrees are still exploited, which leads to inefficiency in real problems.

7 Conclusions

In this paper, we presented a new application of boosting for parse reranking, in which all subtrees are potentially used as distinct features. Although this set-up greatly increases the feature space, the l_1-norm regularization performed by boosting selects a compact and relevant feature set. Our model achieved a comparable or even better accuracy than kernel methods even with an extremely small number of features (subtrees).

References

Kenji Abe, Shinji Kawasoe, Tatsuya Asai, Hiroki Arimura, and Setsuo Arikawa. 2002. Optimized substructure discovery for semi-structured data. In *Proc. of PKDD*, pages 1–14.

Rens Bod. 1998. *Beyond Grammar: An Experience Based Theory of Language*. CSLI Publications/Cambridge University Press.

Rens Bod. 2001. What is the minimal set of fragments that achieves maximal parse accuracy? In *Proc. of ACL*, pages 66–73.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of NAACL*, pages 132–139.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proc. of ACL*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*, pages 175–182.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proc. of ACL*, pages 489–496.

Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proc. of NAACL*, pages 192–199.

Taku Kudo and Yuji Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proc. of EMNLP*, pages 301–308.

Simon Perkins, Kevin Lacker, and James Thiler. 2003. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356.

Gunnar. Rätsch. 2001. *Robust Boosting via Convex Optimization*. Ph.D. thesis, Department of Computer Science, University of Potsdam.

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. 1997. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. of ICML*, pages 322–330.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*, pages 213–220.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proc. of CoNLL-2000 and LLL-2000*, pages 127–132.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Guizhen Yang. 2004. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proc. of SIGKDD*.

Mohammed Zaki. 2002. Efficiently mining frequent trees in a forest. In *Proc. of SIGKDD*, pages 71–80.

Tong Zhang, Fred Damerau, and David Johnson. 2002. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–637.

Automatic Measurement of Syntactic Development in Child Language

Kenji Sagae and **Alon Lavie**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15232
{sagae,alavie}@cs.cmu.edu

Brian MacWhinney
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15232
macw@cmu.edu

Abstract

To facilitate the use of syntactic information in the study of child language acquisition, a coding scheme for Grammatical Relations (GRs) in transcripts of parent-child dialogs has been proposed by Sagae, MacWhinney and Lavie (2004). We discuss the use of current NLP techniques to produce the GRs in this annotation scheme. By using a statistical parser (Charniak, 2000) and memory-based learning tools for classification (Daelemans et al., 2004), we obtain high precision and recall of several GRs. We demonstrate the usefulness of this approach by performing automatic measurements of syntactic development with the Index of Productive Syntax (Scarborough, 1990) at similar levels to what child language researchers compute manually.

1 Introduction

Automatic syntactic analysis of natural language has benefited greatly from statistical and corpus-based approaches in the past decade. The availability of syntactically annotated data has fueled the development of high quality statistical parsers, which have had a large impact in several areas of human language technologies. Similarly, in the study of child language, the availability of large amounts of electronically accessible empirical data in the form of child language transcripts has been shifting much of the research effort towards a corpus-based mentality. However, child language researchers have only recently begun to utilize modern NLP techniques for syntactic analysis. Although it is now common for researchers to rely on automatic morphosyntactic analyses of transcripts to obtain part-of-speech and morphological analyses, their use of syntactic parsing is rare.

Sagae, MacWhinney and Lavie (2004) have proposed a syntactic annotation scheme for the CHILDES database (MacWhinney, 2000), which contains hundreds of megabytes of transcript data and has been used in over 1,500 studies in child language acquisition and developmental language disorders. This annotation scheme focuses on syntactic structures of particular importance in the study of child language. In this paper, we describe the use of existing NLP tools to parse child language transcripts and produce automatically annotated data in the format of the scheme of Sagae et al. We also validate the usefulness of the annotation scheme and our analysis system by applying them towards the practical task of measuring syntactic development in children according to the Index of Productive Syntax, or IPSyn (Scarborough, 1990), which requires syntactic analysis of text and has traditionally been computed manually. Results obtained with current NLP technology are close to what is expected of human performance in IPSyn computations, but there is still room for improvement.

2 The Index of Productive Syntax (IPSyn)

The Index of Productive Syntax (Scarborough, 1990) is a measure of development of child language that provides a numerical score for grammatical complexity. IPSyn was designed for investigating individual differences in child language acquisition, and has been used in numerous studies. It addresses weaknesses in the widely popular Mean Length of Utterance measure, or MLU, with respect to the assessment of development of syntax in children. Because it addresses syntactic structures directly, it has gained popularity in the study of grammatical aspects of child language learning in both research and clinical settings.

After about age 3 (Klee and Fitzgerald, 1985), MLU starts to reach ceiling and fails to properly distinguish between children at different levels of syntactic ability. For these purposes, and because of its higher content validity, IPSyn scores often tells us more than MLU scores. However, the MLU holds the advantage of being far easier to compute. Relatively accurate automated methods for computing the MLU for child language transcripts have been available for several years (MacWhinney, 2000).

Calculation of IPSyn scores requires a corpus of 100 transcribed child utterances, and the identification of 56 specific language structures in each utterance. These structures are counted and used to compute numeric scores for the corpus in four categories (noun phrases, verb phrases, questions and negations, and sentence structures), according to a fixed score sheet. Each structure in the four categories receives a score of zero (if the structure was not found in the corpus), one (if it was found once in the corpus), or two (if it was found two or more times). The scores in each category are added, and the four category scores are added into a final IPSyn score, ranging from zero to 112.[1]

Some of the language structures required in the computation of IPSyn scores (such as the presence of auxiliaries or modals) can be recognized with the use of existing child language analysis tools, such as the morphological analyzer MOR (MacWhinney, 2000) and the part-of-speech tagger POST (Parisse and Le Normand, 2000). However, more complex structures in IPSyn require syntactic analysis that goes beyond what POS taggers can provide. Examples of such structures include the presence of an inverted copula or auxiliary in a wh-question, conjoined clauses, bitransitive predicates, and fronted or center-embedded subordinate clauses.

[1]See (Scarborough, 1990) for a complete listing of targeted structures and the IPSyn score sheet used for calculation of scores.

Sentence (input):
We eat the cheese sandwich

Grammatical Relations (output):

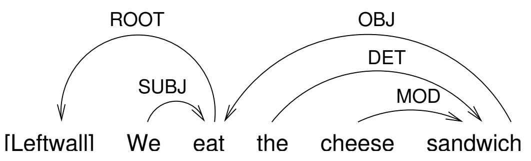


Figure 1: Input sentence and output produced by our system.

3 Automatic Syntactic Analysis of Child Language Transcripts

A necessary step in the automatic computation of IPSyn scores is to produce an automatic syntactic analysis of the transcripts being scored. We have developed a system that parses transcribed child utterances and identifies grammatical relations (GRs) according to the CHILDES syntactic annotation scheme (Sagae et al., 2004). This annotation scheme was designed specifically for child-parent dialogs, and we have found it suitable for the identification of the syntactic structures necessary in the computation of IPSyn.

Our syntactic analysis system takes a sentence and produces a labeled dependency structure representing its grammatical relations. An example of the input and output associated with our system can be seen in figure 1. The specific GRs identified by the system are listed in figure 2.

The three main steps in our GR analysis are: text preprocessing, unlabeled dependency identification, and dependency labeling. In the following subsections, we examine each of them in more detail.

3.1 Text Preprocessing

The CHAT transcription system[2] is the format followed by all transcript data in the CHILDES database, and it is the input format we use for syntactic analysis. CHAT specifies ways of transcribing extra-grammatical material such as disfluency, retracing, and repetition, common in spontaneous spoken language. Transcripts of child language may contain a large amount of extra-grammatical mate-

[2]http://childes.psy.cmu.edu/manuals/CHAT.pdf

SUBJ, ESUBJ, CSUBJ, XSUBJ Subject, expletive subject, clausal subject (finite and non−finite)			**OBJ, OBJ2, IOBJ** Object, second object, indirect object		
COMP, XCOMP Clausal complement (finite and non−finite)			**PRED, CPRED, XPRED** Predicative, clausal predicative (finite and non−finite)		
JCT, CJCT, XJCT Adjunct, clausal adjunct (finite and non−finite)			**MOD, CMOD, XMOD** Nominal modifier, clausal nominal modifier (finite and non−finite)		
AUX Auxiliary	**NEG** Negation	**DET** Determiner	**QUANT** Quantifier	**POBJ** Prepositional object	**PTL** Verb particle
CPZR Complementizer	**COM** Communicator	**INF** Infinitival "to"	**VOC** Vocative	**COORD** Coordinated item	**ROOT** Top node

Figure 2: Grammatical relations in the CHILDES syntactic annotation scheme.

rial that falls outside of the scope of the syntactic annotation system and our GR identifier, since it is already clearly marked in CHAT transcripts. By using the CLAN tools (MacWhinney, 2000), designed to process transcripts in CHAT format, we remove disfluencies, retracings and repetitions from each sentence. Furthermore, we run each sentence through the MOR morphological analyzer (MacWhinney, 2000) and the POST part-of-speech tagger (Parisse and Le Normand, 2000). This results in fairly clean sentences, accompanied by full morphological and part-of-speech analyses.

3.2 Unlabeled Dependency Identification

Once we have isolated the text that should be analyzed in each sentence, we parse it to obtain unlabeled dependencies. Although we ultimately need labeled dependencies, our choice to produce unlabeled structures first (and label them in a later step) is motivated by available resources. Unlabeled dependencies can be readily obtained by processing constituent trees, such as those in the Penn Treebank (Marcus et al., 1993), with a set of rules to determine the lexical heads of constituents. This lexicalization procedure is commonly used in statistical parsing (Collins, 1996) and produces a dependency tree. This dependency extraction procedure from constituent trees gives us a straightforward way to obtain unlabeled dependencies: use an existing statistical parser (Charniak, 2000) trained on the Penn Treebank to produce constituent trees, and extract unlabeled dependencies using the aforementioned head-finding rules.

Our target data (transcribed child language) is from a very different domain than the one of the data used to train the statistical parser (the Wall Street Journal section of the Penn Treebank), but the degradation in the parser's accuracy is acceptable. An evaluation using 2,018 words of in-domain manually annotated dependencies shows that the dependency accuracy of the parser is 90.1% on child language transcripts (compared to over 92% on section 23 of the Wall Street Journal portion of the Penn Treebank). Despite the many differences with respect to the domain of the training data, our domain features sentences that are much shorter (and therefore easier to parse) than those found in Wall Street Journal articles. The average sentence length varies from transcript to transcript, because of factors such as the age and verbal ability of the child, but it is usually less than 15 words.

3.3 Dependency Labeling

After obtaining unlabeled dependencies as described above, we proceed to label those dependencies with the GR labels listed in Figure 2.

Determining the labels of dependencies is in general an easier task than finding unlabeled dependencies in text.[3] Using a classifier, we can choose one of the 30 possible GR labels for each dependency, given a set of features derived from the dependencies. Although we need manually labeled data to train the classifier for labeling dependencies, the size of this training set is far smaller than what would be necessary to train a parser to find labeled dependen-

[3]Klein and Manning (2002) offer an informal argument that constituent labels are much more easily separable in multidimensional space than constituents/distituents. The same argument applies to dependencies and their labels.

cies in one pass.

We use a corpus of about 5,000 words with manually labeled dependencies to train TiMBL (Daelemans et al., 2003), a memory-based learner (set to use the k-nn algorithm with k=1, and gain ratio weighing), to classify each dependency with a GR label. We extract the following features for each dependency:

- The head and dependent words;
- The head and dependent parts-of-speech;
- Whether the dependent comes before or after the head in the sentence;
- How many words apart the dependent is from the head;
- The label of the lowest node in the constituent tree that includes both the head and dependent.

The accuracy of the classifier in labeling dependencies is 91.4% on the same 2,018 words used to evaluate unlabeled accuracy. There is no intersection between the 5,000 words used for training and the 2,018-word test set. Features were tuned on a separate development set of 582 words.

When we combine the unlabeled dependencies obtained with the Charniak parser (and head-finding rules) and the labels obtained with the classifier, overall labeled dependency accuracy is 86.9%, significantly above the results reported (80%) by Sagae et al. (2004) on very similar data.

Certain frequent and easily identifiable GRs, such as DET, POBJ, INF, and NEG were identified with precision and recall above 98%. Among the most difficult GRs to identify were clausal complements COMP and XCOMP, which together amount to less than 4% of the GRs seen the training and test sets. Table 1 shows the precision and recall of GRs of particular interest.

Although not directly comparable, our results are in agreement with state-of-the-art results for other labeled dependency and GR parsers. Nivre (2004) reports a labeled (GR) dependency accuracy of 84.4% on modified Penn Treebank data. Briscoe and Carroll (2002) achieve a 76.5% F-score on a very rich set of GRs in the more heterogeneous and challenging Susanne corpus. Lin (1998) evaluates his MINIPAR system at 83% F-score on identification of GRs, also in data from the Susanne corpus (but using simpler GR set than Briscoe and Carroll).

GR	Precision	Recall	F-score
SUBJ	0.94	0.93	0.93
OBJ	0.83	0.91	0.87
COORD	0.68	0.85	0.75
JCT	0.91	0.82	0.86
MOD	0.79	0.92	0.85
PRED	0.80	0.83	0.81
ROOT	0.91	0.92	0.91
COMP	0.60	0.50	0.54
XCOMP	0.58	0.64	0.61

Table 1: Precision, recall and F-score (harmonic mean) of selected Grammatical Relations.

4 Automating IPSyn

Calculating IPSyn scores manually is a laborious process that involves identifying 56 syntactic structures (or their absence) in a transcript of 100 child utterances. Currently, researchers work with a partially automated process by using transcripts in electronic format and spreadsheets. However, the actual identification of syntactic structures, which accounts for most of the time spent on calculating IPSyn scores, still has to be done manually.

By using part-of-speech and morphological analysis tools, it is possible to narrow down the number of sentences where certain structures may be found. The search for such sentences involves patterns of words and parts-of-speech (POS). Some structures, such as the presence of determiner-noun or determiner-adjective-noun sequences, can be easily identified through the use of simple patterns. Other structures, such as front or center-embedded clauses, pose a greater challenge. Not only are patterns for such structures difficult to craft, they are also usually inaccurate. Patterns that are too general result in too many sentences to be manually examined, but more restrictive patterns may miss sentences where the structures are present, making their identification highly unlikely. Without more syntactic analysis, automatic searching for structures in IPSyn is limited, and computation of IPSyn scores still requires a great deal of manual inspection.

Long, Fey and Channell (2004) have developed a software package, Computerized Profiling (CP), for child language study, which includes a (mostly) automated computation of IPSyn.[4] CP is an extensively developed example of what can be achieved using only POS and morphological analysis. It does well on identifying items in IPSyn categories that do not require deeper syntactic analysis. However, the accuracy of overall scores is not high enough to be considered reliable in practical usage, in particular for older children, whose utterances are longer and more sophisticated syntactically. In practice, researchers usually employ CP as a first pass, and manually correct the automatic output. Section 5 presents an evaluation of the CP version of IPSyn.

Syntactic analysis of transcripts as described in section 3 allows us to go a step further, fully automating IPSyn computations and obtaining a level of reliability comparable to that of human scoring. The ability to search for both grammatical relations and parts-of-speech makes searching both easier and more reliable. As an example, consider the following sentences (keeping in mind that there are no explicit commas in spoken language):

(a) Then [,] he said he ate.
(b) Before [,] he said he ate.
(c) Before he ate [,] he ran.

Sentences (a) and (b) are similar, but (c) is different. If we were looking for a fronted subordinate clause, only (c) would be a match. However, each one of the sentences has an identical part-speech-sequence. If this were an isolated situation, we might attempt to fix it by having tags that explicitly mark verbs that take clausal complements, or by adding lexical constraints to a search over part-of-speech patterns. However, even by modifying this simple example slightly, we find more problems:

(d) Before [,] he told the man he was cold.
(e) Before he told the story [,] he was cold.

Once again, sentences (d) and (e) have identical part-of-speech sequences, but only sentence (e) features a fronted subordinate clause. These limited toy examples only scratch the surface of the difficulties in identifying syntactic structures without syntactic analysis beyond part-of-speech and morphological tagging. In these sentences, searching with GRs is easy: we simply find a GR of clausal type (e.g. CJCT, COMP, CMOD, etc) where the dependent is to the left of its head.

For illustration purposes of how searching for structures in IPSyn is done with GRs, let us look at how to find other IPSyn structures[5]:

- Wh-embedded clauses: search for wh-words whose head, or transitive head (its head's head, or head's head's head...) is a dependent in GR of types [XC]SUBJ, [XC]PRED, [XC]JCT, [XC]MOD, COMP or XCOMP;
- Relative clauses: search for a CMOD where the dependent is to the right of the head;
- Bitransitive predicate: search for a word that is a head of both OBJ and OBJ2 relations.

Although there is still room for under- and over-generalization with search patterns involving GRs, finding appropriate ways to search is often made trivial, or at least much more simple and reliable than searching without GRs. An evaluation of our automated version of IPSyn, which searches for IPSyn structures using POS, morphology and GR information, and a comparison to the CP implementation, which uses only POS and morphology information, is presented in section 5.

5 Evaluation

We evaluate our implementation of IPSyn in two ways. The first is *Point Difference*, which is calculated by taking the (unsigned) difference between scores obtained manually and automatically. The point difference is of great practical value, since it shows exactly how close automatically produced scores are to manually produced scores. The second is *Point-to-Point Accuracy*, which reflects the overall reliability over each individual scoring decision in the computation of IPSyn scores. It is calculated by counting how many decisions (identification of presence/absence of language structures in the transcript being scored) were made correctly, and dividing that

[4] Although CP requires that a few decisions be made manually, such as the disambiguation of the lexical item "'s" as copula vs. genitive case marker, and the definition of sentence breaks for long utterances, the computation of IPSyn scores is automated to a large extent.

[5] More detailed descriptions and examples of each structure are found in (Scarborough, 1990), and are omitted here for space considerations, since the short descriptions are fairly self-explanatory.

number by the total number of decisions. The point-to-point measure is commonly used for assessing the inter-rater reliability of metrics such as the IPSyn. In our case, it allows us to establish the reliability of automatically computed scores against human scoring.

5.1 Test Data

We obtained two sets of transcripts with corresponding IPSyn scoring (total scores, and each individual decision) from two different child language research groups. The first set (A) contains 20 transcripts of children of ages ranging between two and three. The second set (B) contains 25 transcripts of children of ages ranging between eight and nine.

Each transcript in set A was scored fully manually. Researchers looked for each language structure in the IPSyn scoring guide, and recorded its presence in a spreadsheet. In set B, scoring was done in a two-stage process. In the first stage, each transcript was scored automatically by CP. In the second stage, researchers checked each automatic decision made by CP, and corrected any errors manually.

Two transcripts in each set were held out for development and debugging. The final test sets contained: (A) 18 transcripts with a total of 11,704 words and a mean length of utterance of 2.9, and (B) 23 transcripts with a total of 40,819 words and a mean length of utterance of 7.0.

5.2 Results

Scores computed automatically from transcripts parsed as described in section 3 were very close to the scores computed manually. Table 2 shows a summary of the results, according to our two evaluation metrics. Our system is labeled as GR, and manually computed scores are labeled as HUMAN. For comparison purposes, we also show the results of running Long et al.'s automated version of IPSyn, labeled as CP, on the same transcripts.

Point Difference

The average (absolute) point difference between automatically computed scores (GR) and manually computed scores (HUMAN) was 3.3 (the range of HUMAN scores on the data was 21-91). There was no clear trend on whether the difference was positive or negative. In some cases, the automated scores were higher, in other cases lower. The minimum difference was zero, and the maximum difference was 12. Only two scores differed by 10 or more, and 17 scores differed by two or less. The average point difference between HUMAN and the scores obtained with Long et al.'s CP was 8.3. The minimum was zero and the maximum was 21. Sixteen scores differed by 10 or more, and six scores differed by 2 or less. Figure 3 shows the point differences between GR and HUMAN, and CP and HUMAN.

It is interesting to note that the average point differences between GR and HUMAN were similar on sets A and B (3.7 and 2.9, respectively). Despite the difference in age ranges, the two averages were less than one point apart. On the other hand, the average difference between CP and HUMAN was 6.2 on set A, and 10.2 on set B. The larger difference reflects CP's difficulty in scoring transcripts of older children, whose sentences are more syntactically complex, using only POS analysis.

System	Avg. Pt. Difference to HUMAN	Point-to-Point Reliability
GR (Total)	**3.3**	**92.8%**
CP (Total)	8.3	85.4%
GR (Set A)	3.7	92.5%
CP (Set A)	6.2	86.2%
GR (Set B)	2.9	93.0%
CP (Set B)	10.2	84.8%

Table 2: Summary of evaluation results. GR is our implementation of IPSyn based on grammatical relations, CP is Long et al.'s (2004) implementation of IPSyn, and HUMAN is manual scoring.

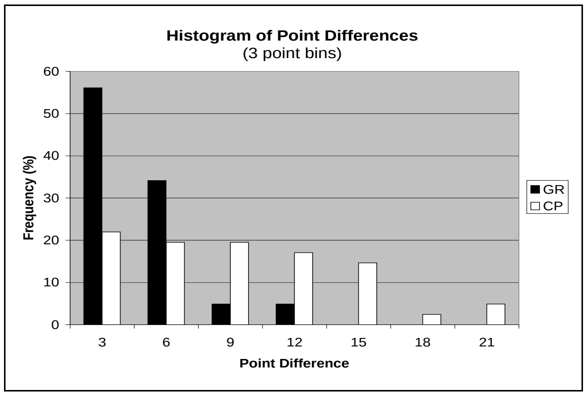


Figure 3: Histogram of point differences between HUMAN scores and GR (black), and CP (white).

Point-to-Point Accuracy

In the original IPSyn reliability study (Scarborough, 1990), point-to-point measurements using 75 transcripts showed the mean inter-rater agreement for IPSyn among human scorers at 94%, with a minimum agreement of 90% of all decisions within a transcript. The lowest agreement between HUMAN and GR scoring for decisions within a transcript was 88.5%, with a mean of 92.8% over the 41 transcripts used in our evaluation. Although comparisons of agreement figures obtained with different sets of transcripts are somewhat coarse-grained, given the variations within children, human scorers and transcript quality, our results are very satisfactory. For direct comparison purposes using the same data, the mean point-to-point accuracy of CP was 85.4% (a relative increase of about 100% in error).

In their separate evaluation of CP, using 30 samples of typically developing children, Long and Channell (2001) found a 90.7% point-to-point accuracy between fully automatic and manually corrected IPSyn scores.[6] However, Long and Channell compared only CP output with manually corrected CP output, while our set A was manually scored from scratch. Furthermore, our set B contained only transcripts from significantly older children (as in our evaluation, Long and Channell observed decreased accuracy of CP's IPSyn with more complex language usage). These differences, and the expected variation from using different transcripts from different sources, account for the difference in our results and Long and Channell's.

5.3 Error Analysis

Although the overall accuracy of our automatically computed scores is in large part comparable to manual IPSyn scoring (and significantly better than the only option currently available for automatic scoring), our system suffers from visible deficiencies in the identification of certain structures within IPSyn.

Four of the 56 structures in IPSyn account for almost half of the number of errors made by our system. Table 3 lists these IPSyn items, with their respective percentages of the total number of errors.

[6]Long and Channell's evaluation also included samples from children with language disorders. Their 30 samples of typically developing children (with a mean age of 5) are more directly comparable to the data used in our evaluation.

IPSyn item	Error
S11 (propositional complement)	16.9%
V15 (copula, modal or aux for emphasis or ellipsis)	12.3%
S16 (relative clause)	10.6%
S14 (bitransitive predicate)	5.8%

Table 3: IPSyn structures where errors occur most frequently, and their percentages of the total number of errors over 41 transcripts.

Errors in items S11 (propositional complements), S16 (relative clauses), and S14 (bitransitive predicates) are caused by erroneous syntactic analyses. For an example of how GR assignments affect IPSyn scoring, let us consider item S11. Searching for the relation COMP is a crucial part in finding propositional complements. However, COMP is one of the GRs that can be identified the least reliably in our set (precision of 0.6 and recall of 0.5, see table 1). As described in section 2, IPSyn requires that we credit zero points to item S11 for no occurrences of propositional complements, one point for a single occurrence, and two points for two or more occurrences. If there are several COMPs in the transcript, we should find about half of them (plus others, in error), and correctly arrive at a credit of two points. However, if there are very few or none, our count is likely to be incorrect.

Most errors in item V15 (emphasis or ellipsis) were caused not by incorrect GR assignments, but by imperfect search patterns. The searching failed to account for a number of configurations of GRs, POS tags and words that indicate that emphasis or ellipsis exists. This reveals another general source of error in our IPSyn implementation: the search patterns that use GR analyzed text to make the actual IPSyn scoring decisions. Although our patterns are far more reliable than what we could expect from POS tags and words alone, these are still hand-crafted rules that need to be debugged and perfected over time. This was the first evaluation of our system, and only a handful of transcripts were used during development. We expect that once child language researchers have had the opportunity to use the system in practical settings, their feedback will allow us to refine the search patterns at a more rapid pace.

6 Conclusion and Future Work

We have presented an automatic way to annotate transcripts of child language with the CHILDES syntactic annotation scheme. By using existing resources and a small amount of annotated data, we achieved state-of-the-art accuracy levels.

GR identification was then used to automate the computation of IPSyn scores to measure grammatical development in children. The reliability of our automatic IPSyn was very close to the inter-rater reliability among human scorers, and far higher than that of the only other computational implementation of IPSyn. This demonstrates the value of automatic GR assignment to child language research.

From the analysis in section 5.3, it is clear that the identification of certain GRs needs to be made more accurately. We intend to annotate more in-domain training data for GR labeling, and we are currently investigating the use of other applicable GR parsing techniques.

Finally, IPSyn score calculation could be made more accurate with the knowledge of the expected levels of precision and recall of automatic assignment of specific GRs. It is our intuition that in a number of cases it would be preferable to trade recall for precision. We are currently working on a framework for soft-labeling of GRs, which will allow us to manipulate the precision/recall trade-off as discussed in (Carroll and Briscoe, 2002).

Acknowledgments

This work was supported in part by the National Science Foundation under grant IIS-0414630.

References

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

John A. Carroll and Edward J. Briscoe. 2002. High precision extraction of grammatical relations. *Proceedings of the 19th International Conference on Computational Linguistics*, (pp. 134-140). Taipei, Taiwan.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. *In Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics.* Seattle, WA.

Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. *Proceedings of the 34th Meeting of the Association for Computational Linguistics* (pp. 184-191). Santa Cruz, CA.

Walter Daelemans, Jacub 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, 2004.

T. Klee and M. D. Fitzgerald. 1985. The relation between grammatical development and mean length of utterance in morphemes. *Journal of Child Language*, 12, 251-269.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 128-135).

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. *In Proceedings of the Workshop on the Evaluation of Parsing Systems.* Granada, Spain.

Steve H. Long and Ron W. Channell. 2001. Accuracy of four language analysis procedures performed automatically. *American Journal of Speech-Language Pathology*, 10(2).

Steven H. Long, Marc E. Fey, and Ron W. Channell. 2004. Computerized Profiling (Version 9.6.0). Cleveland, OH: Case Western Reserve University.

Brian MacWhinney. 2000. The CHILDES Project: Tools for Analyzing Talk. Mahwah, NJ: Lawrence Erlbaum Associates.

Mitchel P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewics. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. *Proceedings of International Conference on Computational Linguistics* (pp. 64-70). Geneva, Switzerland.

Christophe Parisse and Marie-Thrse Le Normand. 2000. Automatic disambiguation of the morphosyntax in spoken language corpora. *Behavior Research Methods, Instruments, and Computers*, 32, 468-481.

Kenji Sagae, Alon Lavie, and Brian MacWhinney. 2004. Adding Syntactic annotations to transcripts of parent-child dialogs. *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004).* Lisbon, Portugal.

Hollis S. Scarborough. 1990. Index of Productive Syntax. *In Applied Psycholinguistics*, 11, 1-22.

Experiments with Interactive Question-Answering

Sanda Harabagiu, Andrew Hickl, John Lehmann, and Dan Moldovan
Language Computer Corporation
Richardson, Texas USA
sanda@languagecomputer.com

Abstract

This paper describes a novel framework for interactive question-answering (Q/A) based on *predictive questioning*. Generated off-line from topic representations of complex scenarios, predictive questions represent requests for information that capture the most salient (and diverse) aspects of a topic. We present experimental results from large user studies (featuring a fully-implemented interactive Q/A system named FERRET) that demonstrates that surprising performance is achieved by integrating predictive questions into the context of a Q/A dialogue.

1 Introduction

In this paper, we propose a new architecture for *interactive* question-answering based on *predictive questioning*. We present experimental results from a currently-implemented interactive Q/A system, named FERRET, that demonstrates that surprising performance is achieved by integrating sources of topic information into the context of a Q/A dialogue.

In interactive Q/A, professional users engage in extended dialogues with automatic Q/A systems in order to obtain information relevant to a complex scenario. Unlike Q/A in isolation, where the performance of a system is evaluated in terms of how well answers returned by a system meet the specific information requirements of a single question, the performance of interactive Q/A systems have traditionally been evaluated by analyzing aspects of the dialogue as a whole. Q/A dialogues have been evaluated in terms of (1) efficiency, defined as the number of questions that the user must pose to find particular information, (2) effectiveness, defined by the relevance of the answers returned, (3) user satisfaction.

In order to maximize performance in these three areas, interactive Q/A systems need a predictive dialogue architecture that enables them to propose related questions about the relevant information that could be returned to a user, given a domain of interest. We argue that interactive Q/A systems depend on three factors: (1) the effective representation of the topic of a dialogue, (2) the dynamic recognition of the structure of the dialogue, and (3) the ability to return relevant answers to a particular question.

In this paper, we describe results from experiments we conducted with our own interactive Q/A system, FERRET, under the auspices of the ARDA AQUAINT[1] program, involving 8 different dialogue scenarios and more than 30 users. The results presented here illustrate the role of predictive questioning in enhancing the performance of Q/A interactions.

In the remainder of this paper, we describe a new architecture for interactive Q/A. Section 2 presents the functionality of several of FERRET's modules and describes the NLP techniques it relies upon. In Section 3, we present one of the dialogue scenarios and the topic representations we have employed. Section 4 highlights the management of the interaction between the user and FERRET, while Section 5 presents the results of evaluating our proposed

[1]AQUAINT is an acronym for *Advanced QUestion Answering for INTelligence.*

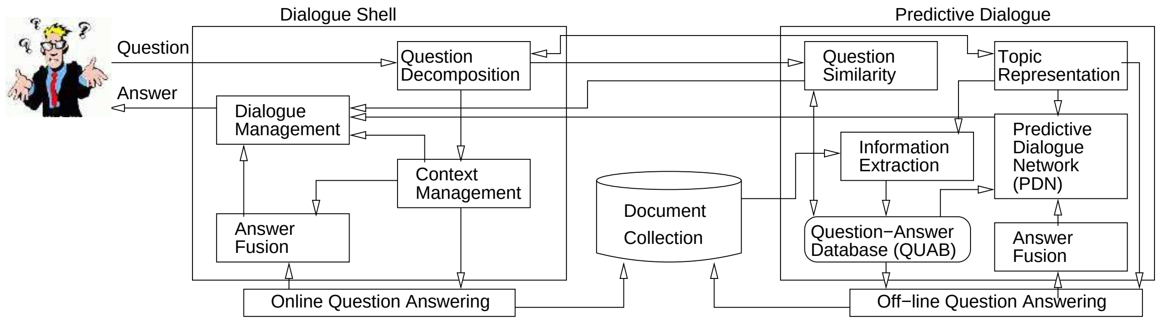


Figure 1: FERRET - A Predictive Interactive Question-Answering Architecture.

model, and Section 6 summarizes the conclusions.

2 Interactive Question-Answering

We have found that the quality of interactions produced by an interactive Q/A system can be greatly enhanced by predicting the range of questions that a user might ask in the context of a given topic. If a large database of topic-relevant questions were available for a wide variety of topics, the accuracy of a state-of-the-art Q/A system such as (Harabagiu et al., 2003) could be enhanced.

In FERRET, our interactive Q/A system, we store such "predicted" pairs of questions and answers in a database known as the *Question Answer Database* (or QUAB). FERRET uses this large set of topic-relevant question-and-answer pairs to improve the interaction with the user by suggesting new questions. For example, when a user asks a question like (Q1) (as illustrated in Table 1), FERRET returns an answer to the question (A1) and proposes (Q2), (Q3), and (Q4) as suggestions of possible continuations of the dialogue. Users then choose how to continue the interaction by either (1) ignoring the suggestions made by the system and proposing a different question, or by (2) selecting one of the proposed questions and examining its answer.

Figure 1 illustrates the architecture of FERRET. The interactions are managed by a *dialogue shell*, which processes questions by transforming them into their corresponding predicate-argument structures[2].

The data collection used in our experiments was made available by the Center for Non-Proliferation Studies (CNS)[3].

(Q1) What weapons are **included in** Egypt's stockpiles?
(A1) The Israelis point to comments made by former President Anwar Sadat, who in 1970 stated that Egypt has biological weapons stored in refrigerators ready to use against Israel if need be. The program might include **"plague, botulism toxin, encephalitis virus, anthrax, Rift Valley fever and mycotoxicosis."**
(Q2) Where did Egypt **inherit** its fi rst stockpiles of chemical weapons?
(Q3) Is there evidence that Egypt has **dismantled** its stockpiles of weapons?
(Q4) Where are Egypt's weapons stockpiles **located**?
(Q5) Who **oversees** Egypt's weapons stockpiles?

Table 1: User question and proposed questions from QUABs

Modules from the FERRET's dialogue shell interact with modules from the predictive dialogue block. Central to the predictive dialogue is the topic representation for each scenario, which enables the population of a Predictive Dialogue Network (PDN). The PDN consists of a large set of questions that were asked or predicted for each topic. It is a network because questions are related by "similarity" links, which are computed by the Question Similarity module. The topic representation enables an Information Extraction module based on (Surdeanu and Harabagiu, 2002) to find topic-relevant information in the document collection and to use it as answers for the QUABs. The questions associated with each predicted answer are generated from patterns that are related to the extraction patterns used for identifying topic relevant information. The quality of the dialog between the user and FERRET depends on the quality of the topic representations and the coverage of the QUABs.

[2]We have employed the same representation of predicate-argument structures as those encoded in PropBank. We use a semantic parser (described in (Surdeanu et al., 2003)) that recognizes predicate-argument structures.

[3]The Center for Non-Proliferation Studies at the Monterrey Institute of International Studies distributes collections of print and online documents on weapons of mass destruction. More information at: http://cns.miis.edu.

GENERAL BACKGROUND

Serving as a background to the scenarios, the following list contains subject areas that may be relevant to the scenarios under examination, and it is provided to assist the analyst in generating questions.
1) Country Profile
2) Government: Type of, Leadership, Relations
3) Military Operations: Army, Navy, Air Force, Leaders, Capabilities, Intentions
4) Allies/Partners: Coalition Forces
5) Weapons: Chemical, Biological, Materials, Stockpiles, Facilities, Access, Research Efforts, Scientists
6) Citizens: Population, Growth Rate, Education
7) Industrial: Major Industrires, Exports, Power Sources
8) Economics: Growth Domestic Product, Growth Rate, Imports
9) Threat Perception: Border and Surrounding States, International, Terrorist Groups
10) Behaviour: Threats, Invasions, Sponsorship and Harboring of Bad Actors
11) Transportation Infrastructure: Kilometers of Road, Rail, Air Runways, Harbors and Ports, Rivers
12) Beliefs: Ideology, Goals, Intentions
13) Leadership:
14) Behaviour: Threats to use WMDs, Actual Usage, Sophistication of Attack, Anectodal or Simultaneous
15) Weapons: Chemical, Bilogical, Materials, Stockpiles, Facilities, Access

SCENARIO: Assessment of Egypt's Biological Weapons

As terrorist Activity in Egypt increases, the Commander of the United States Army believes a better understanding of Egypt's Military capabilities is needed. Egypt's biological weapons database needs to be updated to correspond with the Commander's request. Focus your investigation on Egypt's access to old technology, assistance received from the Soviet Union for development of their pharmaceutical infrastructure, production of toxins and BW agents, stockpiles, exportation of these materials and development technology to Middle Eastern countries, and the effect that this information will have on the United States and Coalition Forces in the Middle East. Please incorporate any other related information to your report.

Figure 2: Example of a Dialogue Scenario.

3 Modeling the Dialogue Topic

Our experiments in interactive Q/A were based on several scenarios that were presented to us as part of the ARDA Metrics Challenge Dialogue Workshop. Figure 2 illustrates one of these scenarios. It is to be noted that the *general background* consists of a list of subject areas, whereas the *scenario* is a narration in which several sub-topics are identified (e.g. *production of toxins* or *exportation of materials*). The creation of scenarios for interactive Q/A requires several different types of domain-specific knowledge and a level of operational expertise not available to most system developers. In addition to identifying a particular *domain of interest*, scenarios must specify the set of relevant *actors*, *outcomes*, and *related topics* that are expected to operate within the domain of interest, the salient *associations* that may exist between entities and events in the scenario, and the specific *timeframe* and *location* that bound the scenario in space and time. In addition, real-world scenarios also need to identify certain operational parameters as well, such as the identity of the scenario's *sponsor* (i.e. the organization sponsoring the research) and *audience* (i.e. the organization receiving the information), as well as a series of *evidence conditions* which specify how much verification information must be subject to before it can be accepted as fact. We assume the set of sub-topics mentioned in the general background and the scenario can be used together to define a topic structure that will govern future interactions with the Q/A system. In order to model this structure, the topic representation that we create considers separate *topic signatures* for each sub-topic.

The notion of topic signatures was first introduced in (Lin and Hovy, 2000). For each subtopic in a scenario, given (a) documents relevant to the sub-topic and (b) documents not relevant to the subtopic, a statistical method based on the likelihood ratio is used to discover a weighted list of the most topic-specific concepts, known as the topic signature. Later work by (Harabagiu, 2004) demonstrated that topic signatures can be further enhanced by discovering the most relevant relations that exist between pairs of concepts. However, both of these types of topic representations are limited by the fact that they require the identification of topic-relevant documents prior to the discovery of the topic signatures. In our experiments, we were only presented with a set of documents relevant to a particular scenario; no further relevance information was provided for individual subject areas or sub-topics.

In order to solve the problem of finding relevant documents for each subtopic, we considered four different approaches:

- **Approach 1**: All documents in the CNS collection were initially clustered using K-Nearest Neighbor (KNN) clustering (Dudani, 1976). Each cluster that contained at least one keyword that described the sub-topic was deemed relevant to the topic.

- **Approach 2**: Since individual documents may contain discourse segments pertaining to different sub-topics, we first used TextTiling (Hearst, 1994) to automatically segment all of the documents in the CNS collection into individual text tiles. These individual discourse segments then served as input to the KNN clustering algorithm described in Approach 1.

- **Approach 3**: In this approach, relevant documents were discovered simultaneously with the discovery of topic signatures. First, we associated a binary *seed relation* r_i for each each sub-topic S_i. (Seed relations were created both by hand and using the method presented in (Harabagiu, 2004).) Since seed relations are by definition relevant to a particular subtopic, they can be used to determine a binary partition of the document collection C into (1) a relevant set of documents R_i (that is, the documents relevant to relation r_i) and (2) a set of non-relevant documents C-R_i. Inspired by the method presented in (Yangarber et al., 2000), a topic signature (as calculated by (Harabagiu, 2004)) is then produced for the set of documents in R_i. For each subtopic S_i defined as part of the dialogue scenario, documents relevant to a corresponding seed relation r_i are added to R iff the relation r_i meets the *density criterion* (as defined in (Yangarber et al., 2000)). If D represents the set of documents where r_i is recognized, then the density criterion can be defined as: $\frac{|D \cap R|}{D \cap C} \gg \frac{|R|}{|C|}$. Once D is added to R_i, then a new topic signature is calculated for R. Relations extracted from the new topic signature can then be used to determine a new document partition by re-iterating the discovery of the topic signature and of the documents relevant to each subtopic.

- **Approach 4**: Approach 4 implements the technique described in Approach 3, but operates at the level of discourse segments (or texttiles) rather than at the level of full documents. As with Approach 2, segments were produced using the TextTiling algorithm.

In modeling the dialogue scenarios, we considered three types of topic-relevant relations: (1) *structural relations*, which represent hypernymy or meronymy relations between topic-relevant concepts, (2) *definition relations*, which uncover the characteristic properties of a concept, and (3) *extraction relations*, which model the most relevant events or states associated with a sub-topic. Although structural relations and definition relations are discovered reliably using patterns available from our Q/A system (Harabagiu et al., 2003), we found only extraction relations to be useful in determining the set of documents relevant to a subtopic. Structural relations were available from concept ontologies implemented in the Q/A system. The definition relations were identified by patterns used for processing definition questions.

Extraction relations are discovered by processing documents in order to identify three types of relations, including: (1) syntactic attachment relations (including subject-verb, object-verb, and verb-PP relations), (2) predicate-argument relations, and (3) salience-based relations that can be used to encode long-distance dependencies between topic-relevant concepts. (Salience-based relations are discovered using a technique first reported in (Harabagiu, 2004) which approximates a Centering Theory-style approach (Kameyama, 1997) to the resolution of coreference.)

Subtopic: Egypt's production of toxins and BW agents

Topic Signature:
produce − phosphorous trichloride (TOXIN)
house − ORGANIZATION
cultivate − non−pathogenic Bacilus Subtilis (TOXIN)
produce − mycotoxins (TOXIN)
acquire − FACILITY

Subtopic: Egypt's allies and partners

Topic Signature:	
provide − COUNTRY	cooperate − COUNTRY
cultivate − COUNTRY	train − PERSON
supply − precursors	supply − know−how

Figure 3: Example of two topic signatures acquired for the scenario illustrated in Figure 2.

We made the extraction relations associated with each topic signature more general (a) by replacing words with their (morphological) root form (e.g. *wounded* with *wound*, *weapons* with *weapon*), (b) by replacing lexemes with their subsuming category from an ontology of 100,000 words (e.g. *truck* is replaced by VEHICLE, ARTIFACT, or OBJECT), and (c) by replacing each name with its name class (*Egypt* with COUNTRY). Figure 3 illustrates the topic signatures resulting for the scenario illustrated in Figure 2.

Once extraction relations were obtained for a particular set of documents, the resulting set of relations were ranked according to a method proposed in (Yangarber, 2003). Under this approach, the score associated with each relation is given by: $score(r) = \frac{Sup(r)}{|D|} * \log_2 Sup(r)$, where $|D|$ represents the cardinality of the documents where the relation is identified, and $Sup(r)$ represents support associated with the relation r. $Sup(r)$ is defined as the sum of the relevance of each document in D: $Sup(r) = \sum_{d \in D} Rel(d)$. The relevance of a document that contains a topic-significant relation can be defined as: $Rel(d) = 1 - \prod_{r \in TS}(1 - Prec(r))$, where TS represents the topic signature of the subtopic[4]. The accuracy of the relation, then, is given by: $Prec(r) = \frac{1}{|D|}(\sum_{d \in D} Rel^{s_i}(d) - \sum_{j \neq i} Rel^{s_j}(d))$. Here, $Rel^{S_i}(d)$ measures the relevance of a subtopic S_i to a particular document d, while $Rel^{S_j}(d)$ measures the relevance of d to another subtopic, S_j.

We use a different learner for each subtopic in order to train simultaneously on each iteration. (The calculation of topic signatures continues to iterate until there are no more relations that can be added to the overall topic signature.) When the precision of a relation to a subtopic S_i is computed, it takes into account the *negative* evidence of its relevance to any other subtopic $S_i \neq S_j$. If $Prec(r) \leq 0$, the relation is not included in the topic signature, where relations are ranked by the score $Score(r) = Prec(r) * log(Sup(r))$.

Representing topics in terms of relevant concepts and relations is important for the processing of questions asked within the context of a given topic. For interactive Q/A, however, the ideal topic-structured representation would be in the form of question-answer pairs (QUABs) that model the individual segments of the scenario. We have currently created two sets of QUABs: a handcrafted set and an automatically-generated set. For the manually-created set of QUABs, 4 linguists manually generated 3210 question-answer pairs for each of the 8 dialogue scenarios considered in our experiments.

In a separate effort, we devised a process for automatically populating the QUAB for each scenario. In order to generate question-answer pairs for each subtopic, we first identified relevant text passages in the document collection to serve as "answers" and then generated individual questions that could be answered by each answer passage.

[4]Initially, TS contains only the seed relation. Additional relations can be added with each iteration.

◇ **Answer Identification**: We defined an *answer passage* as a contiguous sequence of sentences with a positive *answer rank* and a *passage price* of ≤ 4. To select answer passages for each subtopic S_i, we calculate an *answer rank*, $rank(a) = \sum_{r_i} score(r_i)$, that sums across the scores of each relation from the topic signature that is identified in the same text window. Initially, the text window is set to one sentence. (If the sentence is part of a quote, however, the text window is immediately expanded to encompass the entire sentence that contains the quote.) Each passage with $rank(a) > 0$ is then considered to be a *candidate answer passage*. The text window of each candidate answer passage is then expanded to include the following sentence. If the answer rank does not increase with the addition of the succeeding sentence, then the *price* (p) of the candidate answer passage is incremented by 1, otherwise it is decremented by 1. The text window of each candidate answer passage continues to expand until $p = 4$. Before the ranked list of candidate answers can be considered by the Question Generation module, answer passages with a positive price p are stripped of the last p sentences.

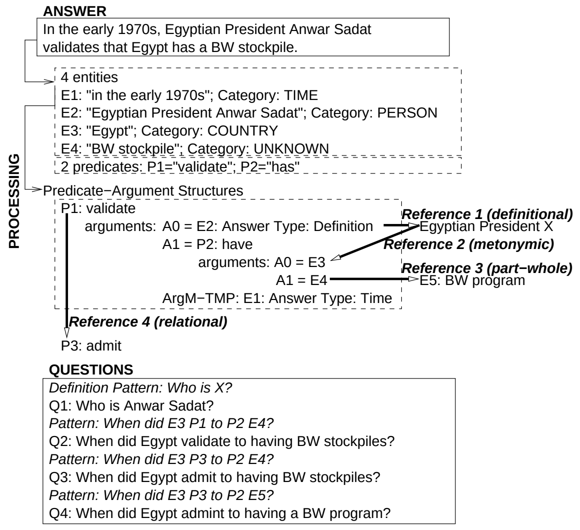
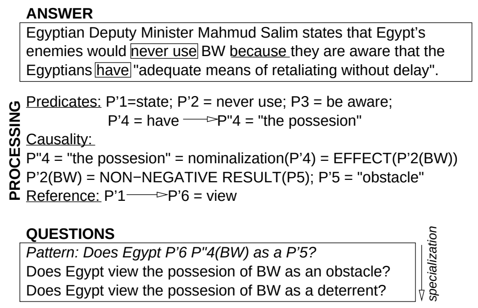

ANSWER
In the early 1970s, Egyptian President Anwar Sadat validates that Egypt has a BW stockpile.
PROCESSING
4 entities
E1: "in the early 1970s"; Category: TIME
E2: "Egyptian President Anwar Sadat"; Category: PERSON
E3: "Egypt"; Category: COUNTRY
E4: "BW stockpile"; Category: UNKNOWN
2 predicates: P1="validate"; P2="has"
Predicate−Argument Structures
P1: validate
arguments: A0 = E2: Answer Type: Definition
A1 = P2: have
arguments: A0 = E3
A1 = E4
ArgM−TMP: E1: Answer Type: Time
Reference 1 (definitional)
Egyptian President X
Reference 2 (metonymic)
Reference 3 (part−whole)
E5: BW program
Reference 4 (relational)
P3: admit
QUESTIONS
Definition Pattern: Who is X?
Q1: Who is Anwar Sadat?
Pattern: When did E3 P1 to P2 E4?
Q2: When did Egypt validate to having BW stockpiles?
Pattern: When did E3 P3 to P2 E4?
Q3: When did Egypt admit to having BW stockpiles?
Pattern: When did E3 P3 to P2 E5?
Q4: When did Egypt admint to having a BW program?

Figure 4: Associating Questions with Answers.

◇ **Question Generation**: In order to automatically generate questions from answer passages, we considered the following two problems:

- **Problem 1**: Every word in an answer passage can refer to an entity, a relation, or an event. In order for question generation be successful, we must determine whether a particular reference is “interesting” enough to the scenario such that it deserves to be mentioned in a topic-relevant question. For example, Figure 4 illustrates an answer that includes two predicates and four entities. In this case, four types of reference are used to associate these linguistic objects with other related objects: (a) *definitional reference*, used to link entity (E1) “*Anwar Sadat*” to a corresponding attribute “*Egyptian President*”, (b) *metonymic reference*, since (E1) can be coerced into (E2), (c) *part-whole reference*, since “*BW stockpiles*”(E4) necessarily imply the existence of a “*BW program*”(E5), and (d) *relational reference*, since *validating* is subsumed as part of the meaning of *declaring* (as determined by WordNet glosses), while *admitting* can be defined in terms of *declaring*, as in *declaring [to be true]*.


ANSWER

Egyptian Deputy Minister Mahmud Salim states that Egypt’s enemies would never use BW because they are aware that the Egyptians have "adequate means of retaliating without delay".

PROCESSING

Predicates: P’1=state; P’2 = never use; P3 = be aware;
P’4 = have ⟶ P"4 = "the possesion"

Causality:
P"4 = "the possesion" = nominalization(P’4) = EFFECT(P’2(BW))
P’2(BW) = NON−NEGATIVE RESULT(P5); P’5 = "obstacle"

Reference: P’1 ⟶ P’6 = view

QUESTIONS

Pattern: Does Egypt P’6 P"4(BW) as a P’5?
Does Egypt view the possesion of BW as an obstacle?
Does Egypt view the possesion of BW as a deterrent?

Figure 5: Questions for Implied Causal Relations.

- **Problem 2**: We have found that the identification of the association between a candidate answer and a question depends on (a) the recognition of predicates and entities based on both the output of a named entity recognizer and a semantic parser (Surdeanu et al., 2003) and their structuring into predicate-argument frames, (b) the resolution of reference (addressed in Problem 1), (c) the recognition of implicit relations between predications stated in the answer. Some of these implicit relations are referential, as is the relation between predicates P_1 and P_3 illustrated in Figure 4. A special case of implicit relations are the causal relations. Figure 5 illustrates an answer where a causal relation exists and is marked by the cue phrase *because*. Predicates – like those in Figure 5 – can be phrasal (like P_3') or negative (like P_2'). Causality is established between predicates P_2' and P_4’ as they are the ones that ultimately determine the selection of the answer. The predicate p_4' can be substituted by its nominalization since Arg_1 of P_2 is *BW*, the same argument is transferred to P_4''. The causality implied by the answer from Figure 5 has two components: (1) the effect (i.e. the predicate P_4'') and (2) the result, which eliminates the semantic effect of the negative polarity item *never* by implying the predicate p_5, *obstacle*. The questions that are generated are based on question patterns associated with causal relations and therefore allow different degrees for the specificity of the resultative, i.e *obstacle* or *deterrent*.

We generated several questions for each answer passage. Questions were generated based on patterns that were acquired to model interrogations using relations between predicates and their arguments. Such interrogations are based on (1) associations between the answer type (e.g. DATE) and the question stem (e.g. “*when*” and (2) the relation between predicates, question stem and the words that determine the answer type (Narayanan and Harabagiu, 2004). In order to obtain these predicate-argument patterns, we used 30% (approximately 1500 questions) of the handcrafted question-answer pairs, selected at random from each of the 8 dialogue scenarios. As Figures 4 and 5 illustrate, we used patterns based on (a) embedded predicates and (b) causal or counterfactual predicates.

4 Managing Interactive Q/A Dialogues

As illustrated in Figure 1, the main idea of managing dialogues in which interactions with the Q/A system occur is based on the notion of predictions, i.e. by proposing to the user a small set of questions that tackle the same subject as her question (as illustrated in Table 1). The advantage is that the user can follow-up with one of the pre-processed questions, that has a correct answer and resides in one of the QUABs. This enhances the effectiveness of the dialogue. It also may impact on the efficiency, i.e. the number of questions being asked if the QUABs have good coverage of the subject areas of the scenario. Moreover, complex questions, that generally are not processed with high accuracy by current state-of-the-art Q/A systems, are associated with predictive questions that represent decompositions based on similarities between predicates and arguments of the original question and the predicted questions.

The selection of the questions from the QUABs that are proposed for each user question is based on a similarity-metric that ranks the QUAB questions. To compute the similarity metric, we have experimented with seven different metrics. The first four metrics were introduced in (Lytinen and Tomuro, 2002).

- **Similarity Metric 1** is based on two processing steps:
  (a) the content words of the questions are weighted using the $tfidf$ measure used in Information Retrieval $w_i = w(t_i) = (1 + \log(tf_i))\frac{\log N}{df_i}$, where N is the number of questions in the QUAB, df_i is the number of questions containing t_i and tf_i is the number of times t_i appears in the question. This allows the user question and any QUAB question to be transformed into two vectors, $v_u = \langle w_{u_1}, w_{u_2}, ..., w_{u_n}\rangle$ and $v_q = \langle w_{q_1}, w_{q_2}, ..., w_{q_m}\rangle$;
  (b) the term vector similarity is used to compute the similarity between the user question and any question from the QUAB: $\cos(v_u, v_q) = (\sum_i w_{u_i} w_{q_i})/((\sum_i w_{u_i}^2)^{\frac{1}{2}} \times (\sum_i w_{q_i}^2)^{\frac{1}{2}})$

- **Similarity Metric 2** is based on the percent of user question terms that appear in the QUAB question. It is obtained by finding the intersection of the terms in the term vectors of the two questions.

- **Similarity Metric 3** is based on semantic information available from WordNet. It involves:
  (a) finding the minimum path between WordNet concepts. Given two terms t_1 and t_2, each with n and m WordNet senses $S_1 = \{s_1, ..., s_n\}$ and $S_2 = \{r_1, ..., r_m\}$. The semantic distance between the terms $\delta(t_1, t_2)$ is defined by the minimum of all the possible pairwise semantic distances between S_1 and S_2: $\delta(t_1, t_2) = \min_{s_i \in S_1, r_j \in S_2} D(s_i, r_j)$, where $D(s_i, r_j)$ is the path length between s_i and r_j.
  (b) the semantic similarity between the user question $T_u = \langle u_1, u_2, ..., u_n\rangle$ and the QUAB question $T_q = \langle b_1, b_2, ..., b_m\rangle$ to be defined as $sem(T_u, T_q) = \frac{I(T_u,T_q)+I(T_q,T_u)}{|T_u|+|T_q|}$, where $I(T_x, T_y) = \sum_{x \in T_x} \frac{1}{1+\min_{y \in T_y} \delta(x,y)}$

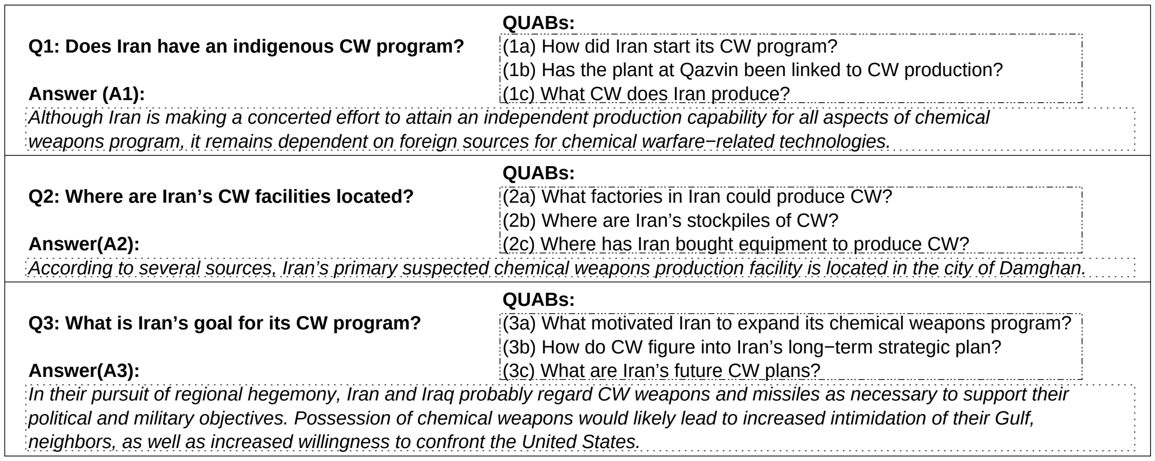
- **Similarity Metric 4** is based on the question type similarity. Instead of using the question class, determined by its stem, whenever we could recognize the answer type expected by the question, we used it for matching. As backoff only, we used a question type similarity based on a matrix akin to the one reported in (Lytinen and Tomuro, 2002)

- **Similarity Metric 5** is based on question concepts rather than question terms. In order to translate question terms into concepts, we replaced (a) *question stems* (i.e. a *WH*-word + NP construction) with expected answer types (taken from the answer type hierarchy employed by FERRET's Q/A system) and (b) *named entities* with corresponding their corresponding classes. Remaining nouns and verbs were also replaced with their WordNet semantic classes, as well. Each concept was then associated with a weight: concepts derived from named entities classes were weighted heavier than concepts from answer types, which were in turn weighted heavier than concepts taken from WordNet clases. Similarity was then computed across "matching" concepts. [5] The resultant similarity score was based on three variables:
  x = sum of the weights of all concepts matched between a *user query* (Q_u) and a *QUAB query* (Q_b);
  y = sum of the weights of all unmatched concepts in Q_u;
  z = sum of the weights of all unmatched concepts in Q_b;
  The similarity between Q_u and Q_b was calculated as $x - (p_u \times y) - (p_b \times z)$, where p_u and p_b were used as coefficients to penalize the contribution of unmatched concepts in Q_u and Q_b respectively. [6]

- **Similarity Metric 6** is based on the fact that the

[5]In the case of ambiguous nouns and verbs associated with multiple WordNet classes, all possible classes for a term were considered in matching.

[6]We set p_u = 0.4 and p_b = 0.1 in our experiments.



QUABs:

Q1: Does Iran have an indigenous CW program?

(1a) How did Iran start its CW program?
(1b) Has the plant at Qazvin been linked to CW production?
(1c) What CW does Iran produce?

Answer (A1):

Although Iran is making a concerted effort to attain an independent production capability for all aspects of chemical weapons program, it remains dependent on foreign sources for chemical warfare−related technologies.

QUABs:

Q2: Where are Iran's CW facilities located?

(2a) What factories in Iran could produce CW?
(2b) Where are Iran's stockpiles of CW?
(2c) Where has Iran bought equipment to produce CW?

Answer(A2):

According to several sources, Iran's primary suspected chemical weapons production facility is located in the city of Damghan.

QUABs:

Q3: What is Iran's goal for its CW program?

(3a) What motivated Iran to expand its chemical weapons program?
(3b) How do CW figure into Iran's long−term strategic plan?
(3c) What are Iran's future CW plans?

Answer(A3):

In their pursuit of regional hegemony, Iran and Iraq probably regard CW weapons and missiles as necessary to support their political and military objectives. Possession of chemical weapons would likely lead to increased intimidation of their Gulf, neighbors, as well as increased willingness to confront the United States.

Figure 6: A sample interactive Q/A dialogue.

QUAB questions are clustered based on their mapping to a vector of important concepts in the QUAB.The clustering was done using the K-Nearest Neighbor (KNN) method (Dudani, 1976). Instead of measuring the similarity between the user question and each question in the QUAB, similarities are computed only between the user question and the centroid of each cluster.

- **Similarity Metric 7** was derived from the results of Similarity Metrics 5 and 6 above. In this case, if the QUAB question (Q_b) that was deemed to be most similar to a user question (Q_u) under Similarity Metric 5 is contained in the cluster of QUAB questions deemed to be most similar to Q_u under Similarity Metric 6, then Q_b receives a *cluster adjustment score* in order to boost its ranking within its QUAB cluster. We calculate the cluster adjustment score as $score_{adj}(Q_b) = (sim_5 * (1 - C_f)) + (sim_6 * C_f)$, where C_f represents the difference in rank between the centroid of the cluster and the previous rank of the QUAB question Q_b.

In the currently-implemented version of FERRET, we used Similarity Metric 5 to automatically identify the set of 10 QUAB questions that were most similar to a user's question. These question-and-answer pairs were then returned to the user – along with answers from FERRET's automatic Q/A system – as potential continuations of the Q/A dialogue. We used the remaining 6 similarity metrics described in this section to manually assess the impact of similarity on a Q/A dialogue.

5 Experiments with Interactive Q/A Dialogues

To date, we have used FERRET to produce over 90 Q/A dialogues with human users. Figure 6 illustrates three turns from a real dialogue from a human user investigating Iran's chemical weapons prorgram. As it can be seen coherence can be established between the user's questions and the system's answers (e.g. Q3 is related to both A1 and A3) as well as between the QUABs and the user's follow-up questions (e.g. QUAB (1b) is more related to Q2 than either Q1 or A1). Coherence alone is not sufficient to analyze the quality of interactions, however.

In order to better understand interactive Q/A dialogues, we have conducted three sets of experiments with human users of FERRET. In these experiments, users were allotted two hours to interact with Ferret to gather information requested by a dialogue scenario similar to the one presented in Figure 2. In Experiment 1 (E1), 8 U.S. Navy Reserve (USNR) intelligence analysts used FERRET to research 8 different scenarios related to chemical and biological weapons. Experiment 2 and Experiment 3 considered several of the same scenarios addressed in E1: E2 included 24 mixed teams of analysts and novice users working with 2 scenarios, while E3 featured 4 USNR analysts working with 6 of the original 8 scenarios. (Details for each experiment are provided in Table 2.) Users were also given a task to focus their research; in E1 and E3, users prepared a short report detailing their findings; in E2, users were given a list of "challenge" questions to answer.

Exp	Users	QUABs?	Scenarios	Topics
E1	8	Yes	8	Egypt BW, Russia CW, South Africa CW, India CW, North Korea CBW, Pakistan CW, Libya CW, Iran CW
E2	24	Yes	2	Egypt BW, Russia CW
E3	4	No	6	Egypt BW, Russia CW, North Korea CBW, Pakistan CW India CW, Libya CW, Iran CW

Table 2: Experiment details

In E1 and E2, users had access to a total of 3210 QUAB questions that had been hand-created by developers for each the 8 dialogue scenarios. (Table 3 provides totals for each scenario.) In E3, users performed research with a version of FERRET that included no QUABs at all.

Scenario	Handcrafted QUABs
INDIA	460
LIBYA	414
IRAN	522
NORTH KOREA	316
PAKISTAN	322
SOUTH AFRICA	454
RUSSIA	366
EGYPT	356
Testing Total	3210

Table 3: QUAB distribution over scenarios

We have evaluated FERRET by measuring efficiency, effectiveness, and user satisfaction:

Efficiency FERRET's QUAB collection enabled users in our experiments to find more relevant information by asking fewer questions. When manually-created QUABs were available (E1 and E2), users submitted an average of 12.25 questions each session. When no QUABs were available (E3), users entered a total of 44.5 questions per session. Table 4 lists the number of QUAB question-answer pairs selected by users and the number of user questions entered by users during the 8 scenarios considered in E1. In E2, freed from the task of writing a research report, users asked significantly ($p < 0.05$) fewer questions and selected fewer QUABs than they did in E1. (See Table 5).

Effectiveness QUAB question-answer pairs also improved the overall accuracy of the answers returned by FERRET. To measure the effectiveness of a Q/A dialogue, human annotators were used to perform a post-hoc analysis of how relevant the QUAB pairs returned by FERRET were to each question

Country	n	QUAB (avg.)	User Q (avg.)	Total (avg.)
India	2	21.5	13.0	34.5
Libya	2	12.0	9.0	21.0
Iran	2	18.5	11.0	29.5
N.Korea	2	16.5	7.5	34.0
Pakistan	2	29.5	15.5	45.0
S.Africa	2	14.5	6.0	20.5
Russia	2	13.5	15.5	29.0
Egypt	2	15.0	20.5	35.5
TOTAL(E1)	16	17.63	12.25	29.88

Table 4: Efficiency of Dialogues in Experiment 1

Country	n	QUAB (avg.)	User Q (avg.)	Total (avg.)
Russia	24	8.2	5.5	13.7
Egypt	24	10.8	7.6	18.4
TOTAL(E2)	48	9.50	6.55	16.05

Table 5: Efficiency of Dialogues in Experiment 2

entered by a user: each QUAB pair returned was graded as "relevant" or "irrelevant" to a user question in a forced-choice task. Aggregate relevance scores were used to calculate (1) the percentage of relevant QUAB pairs returned and (2) the mean reciprocal rank (MRR) for each user question. MRR is defined as $\frac{1}{n}\sum_{i=1}\frac{1}{r_i}$, whree r_i is the lowest rank of any relevant answer for the i^{th} user query[7]. Table 6 describes the performance of FERRET when each of the 7 similarity measures presented in Section 4 are used to return QUAB pairs in response to a query. When only answers from FERRET's automatic Q/A system were available to users, only 15.7% of system responses were deemed to be relevant to a user's query. In contrast, when manually-generated QUAB pairs were introduced, as high as 84% of the system's responses were deemed to be relevant. The results listed in Table 6 show that the best metric is Similarity Metric 5. Thse results suggest that the selection of relevant questions depends on sophisticated similarity measures that rely on conceptual hierarchies and semantic recognizers.

We evaluated the quality of each of the four sets of automatically-generated QUABs in a similar fashion. For each question submitted by a user in E1, E2, and E3, we collected the top 5 QUAB question-answer pairs (as determined by Similarity Metric 5) that FERRET returned. As with the manually-generated QUABs, the automatically-

[7]We chose MRR as our scoring metric because it reflects the fact that a user is most likely to examine the first few answers from any system, but that all correct answers returned by the system have some value because users will sometimes examine a very large list of query results.

	% of Top 5 Responses Relevant to User Q	% of Top 1 Responses Relevant to User Q	MRR
Without QUAB	15.73%	26.85%	0.325
Similarity 1	82.61%	60.63%	0.703
Similarity 2	79.95%	58.45%	0.681
Similarity 3	79.47%	56.04%	0.664
Similarity 4	78.26%	46.14%	0.592
Similarity 5	**84.06%**	**68.36%**	**0.753**
Similarity 6	81.64%	56.04%	0.671
Similarity 7	84.54%	64.01%	0.730

Table 6: Effectiveness of dialogs

generated pairs were submitted to human assessors who annotated each as "relevant" or irrelevant to the user's query. Aggregate scores are presented in Table 7.

	Egypt		Russia	
Approach	% of Top 5 Responses Rel. to User Q	MRR	% of Top 5 Responses Rel. to User Q	MRR
Approach 1	40.01%	0.295	60.25%	0.310
Approach 2	36.00%	0.243	72.00%	0.475
Approach 3	44.62%	0.271	60.00%	0.297
Approach 4	68.05%	0.510	68.00%	0.406

Table 7: Quality of QUABs acquired automatically

User Satisfaction Users were consistently satisfied with their interactions with FERRET. In all three experiments, respondents claimed that they found that FERRET (1) gave meaningful answers, (2) provided useful suggestions, (3) helped answer specific questions, and (4) promoted their general understanding of the issues considered in the scenario. Complete results of this study are presented in Table 8[8].

Factor	E1	E2	E3
Promoted understanding	3.40	3.20	3.75
Helped with specifi c questions	3.70	3.60	3.25
Make good use of questions	3.40	3.55	3.0
Gave new scenario insights	3.00	3.10	2.2
Gave good collection coverage	3.75	3.70	3.75
Stimulated user thinking	3.50	3.20	2.75
Easy to use	3.50	3.55	4.10
Expanded understanding	3.40	3.20	3.00
Gave meaningful answers	4.10	3.60	2.75
Was helpful	4.00	3.75	3.25
Helped with new search methods	2.75	3.05	2.25
Provided novel suggestions	3.25	3.40	2.65
Is ready for work environment	2.85	2.80	3.25
Would speed up work	3.25	3.25	3.00
Overall like of system	3.75	3.60	3.75

Table 8: User Satisfaction Survey Results

6 Conclusions

We believe that the quality of Q/A interactions depends on the modeling of scenario topics. An ideal model is provided by question-answer databases (QUABs) that are created off-line and then used to make suggestions to a user of potential relevant continuations of a discourse. In this paper, we have presented FERRET, an interactive Q/A system which makes use of a novel Q/A architecture that integrates QUAB question-answer pairs into the processing of questions. Experiments with FERRET have shown that, in addition to being rapidly adopted by users as valid suggestions, the incorporation of QUABs into Q/A can greatly improve the overall accuracy of an interactive Q/A dialogue.

[8]Evaluation scale: 1-does not describe the system, 5-completely describes the system

References

S. Dudani. 1976. The distance-weighted k-nearest-neighbour rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327.

S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, J. Williams, and J. Bensley. 2003. Answer Mining by Combining Extraction Techniques with Abductive Reasoning. In *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003)*.

Sanda Harabagiu. 2004. Incremental Topic Representations. In *Proceedings of the 20th COLING Conference*, Geneva, Switzerland.

Marti Hearst. 1994. Multi-Paragraph Segmentation of Expository Text. In *Proceedings of the 32nd Meeting of the Association for Computational Linguistics*, pages 9–16.

Megumi Kameyama. 1997. Recognizing Referential Links: An Information Extraction Perspective. In *Workshop of Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts, (ACL-97/EACL-97)*, pages 46–53.

Chin-Yew Lin and Eduard Hovy. 2000. The Automated Acquisition of Topic Signatures for Text Summarization. In *Proceedings of the 18th COLING Conference*, pages 495–501.

S. Lytinen and N. Tomuro. 2002. The Use of Question Types to Match Questions in FAQFinder. In *Papers from the 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, pages 46–53.

Srini Narayanan and Sanda Harabagiu. 2004. Question Answering Based on Semantic Structures. In *Proceedings of the 20th COLING Conference*, Geneva, Switzerland.

Mihai Surdeanu and Sanda M. Harabagiu. 2002. Infratructure for open-domanin information extraction. In *Conference for Human Language Technology (HLT-2002)*.

Mihai Surdeanu, Sanda M. Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *ACL*, pages 8–15.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the 18th COLING Conference*, pages 940–946.

Roman Yangarber. 2003. Counter-Training in Discovery of Semantic Patterns. In *Proceedings of the 41th Meeting of the Association for Computational Linguistics*, pages 343–350.

Question Answering as Question-Biased Term Extraction: A New Approach toward Multilingual QA

Yutaka Sasaki
Department of Natural Language Processing
ATR Spoken Language Communication Research Laboratories
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0288 Japan
yutaka.sasaki@atr.jp

Abstract

This paper regards *Question Answering (QA)* as *Question-Biased Term Extraction (QBTE)*. This new QBTE approach liberates QA systems from the heavy burden imposed by *question types* (or *answer types*). In conventional approaches, a QA system analyzes a given question and determines the question type, and then it selects answers from among answer candidates that match the question type. Consequently, the output of a QA system is restricted by the design of the question types. The QBTE directly extracts answers as terms biased by the question. To confirm the feasibility of our QBTE approach, we conducted experiments on the CRL QA Data based on 10-fold cross validation, using *Maximum Entropy Models (MEMs)* as an ML technique. Experimental results showed that the trained system achieved 0.36 in MRR and 0.47 in Top5 accuracy.

1 Introduction

The conventional *Question Answering (QA)* architecture is a cascade of the following building blocks:

Question Analyzer analyzes a question sentence and identifies the *question types* (or *answer types*).

Document Retriever retrieves documents related to the question from a large-scale document set.

Answer Candidate Extractor extracts answer candidates that match the question types from the retrieved documents.

Answer Selector ranks the answer candidates according to the syntactic and semantic conformity of each answer with the question and its context in the document.

Typically, question types consist of *named entities*, e.g., PERSON, DATE, and ORGANIZATION, *numerical expressions*, e.g., LENGTH, WEIGHT, SPEED, and *class names*, e.g., FLOWER, BIRD, and FOOD. The question type is also used for selecting answer candidates. For example, if the question type of a given question is PERSON, the answer candidate extractor lists only person names that are tagged as the named entity PERSON.

The conventional QA architecture has a drawback in that the question-type system restricts the range of questions that can be answered by the system. It is thus problematic for QA system developers to carefully design and build an answer candidate extractor that works well in conjunction with the question-type system. This problem is particularly difficult when the task is to develop a multilingual QA system to handle languages that are unfamiliar to the developer. Developing high-quality tools that can extract named entities, numerical expressions, and class names for each foreign language is very costly and time-consuming.

Recently, some pioneering studies have investigated approaches to automatically construct QA components from scratch by applying machine learning techniques to training data (Ittycheriah et al., 2001a)(Ittycheriah et al., 2001b)(Ng et al., 2001) (Pasca and Harabagiu)(Suzuki et al., 2002)(Suzuki

Table 1: Number of Questions in Question Types of CRL QA Data

# of Questions	# of Question Types	Example
1-9	74	AWARD, CRIME, OFFENSE
10-50	32	PERCENT, N_PRODUCT, YEAR_PERIOD
51-100	6	COUNTRY, COMPANY, GROUP
100-300	3	PERSON, DATE, MONEY
Total	115	

et al., 2003) (Zukerman and Horvitz, 2001)(Sasaki et al., 2004). These approaches still suffer from the problem of preparing an adequate amount of training data specifically designed for a particular QA system because each QA system uses its own question-type system. It is very typical in the course of system development to redesign the question-type system in order to improve system performance. This inevitably leads to revision of a large-scale training dataset, which requires a heavy workload.

For example, assume that you have to develop a Chinese or Greek QA system and have 10,000 pairs of question and answers. You have to manually classify the questions according to your own question-type system. In addition, you have to annotate the tags of the question types to large-scale Chinese or Greek documents. If you wanted to redesign the question type ORGANIZATION to three categories, COMPANY, SCHOOL, and OTHER_ORGANIZATION, then the ORGANIZATION tags in the annotated document set would need to be manually revisited and revised.

To solve this problem, this paper regards Question Answering as *Question-Biased Term Extraction (QBTE)*. This new QBTE approach liberates QA systems from the heavy burden imposed by question types.

Since it is a challenging as well as a very complex and sensitive problem to directly extract answers without using question types and only using features of questions, correct answers, and contexts in documents, we have to investigate the feasibility of this approach: how well can answer candidates be extracted, and how well are answer candidates ranked?

In response, this paper employs the machine learning technique *Maximum Entropy Models (MEMs)* to extract answers to a question from documents based on question features, document features, and the combined features. Experimental results show the performance of a QA system that applies MEMs.

2 Preparation

2.1 Training Data

Document Set Japanese newspaper articles of The Mainichi Newspaper published in 1995.

Question/Answer Set We used the CRL[1] QA Data (Sekine et al., 2002). This dataset comprises 2,000 Japanese questions with correct answers as well as question types and IDs of articles that contain the answers. Each question is categorized as one of 115 hierarchically classified question types.

The document set is used not only in the training phase but also in the execution phrase.

Although the CRL QA Data contains question types, *the information of question types are not used for the training*. This is because more than the 60% of question types have fewer than 10 questions as examples (Table 1). This means it is very unlikely that we can train a QA system that can handle this 60% due to data sparseness. [2] Only for the purpose of analyzing experimental results in this paper do we refer to the question types of the dataset.

2.2 Learning with Maximum Entropy Models

This section briefly introduces the machine learning technique Maximum Entropy Models and describes how to apply MEMs to QA tasks.

2.2.1 Maximum Entropy Models

Let $\mathcal{X}$ be a set of input symbols and $\mathcal{Y}$ be a set of class labels. A sample (x, y) is a pair of input x=$\{x_1,\ldots,x_m\}$ ($x_i \in \mathcal{X}$) and output $y \in \mathcal{Y}$.

[1]Presently, National Institute of Information and Communications Technology (NICT), Japan

[2]A machine learning approach to hierarchical question analysis was reported in (Suzuki et al., 2003), but training and maintaining an answer extractor for question types of fine granularity is not an easy task.

The Maximum Entropy Principle (Berger et al., 1996) is to find a model $p* = \underset{p \in C}{\operatorname{argmax}} H(p)$, which means a probability model $p(y|x)$ that maximizes entropy $H(p)$.

Given data $(x^{(1)}, y^{(1)})$,...,$(x^{(n)}, y^{(n)})$, let $\bigcup_k (x^{(k)} \times \{y^{(k)}\}) = \{\langle \tilde{x}_1, \tilde{y}_1 \rangle, ..., \langle \tilde{x}_i, \tilde{y}_i \rangle, ..., \langle \tilde{x}_m, \tilde{y}_m \rangle\}$. This means that we enumerate all pairs of an input symbol and label and represent them as $\langle \tilde{x}_i, \tilde{y}_i \rangle$ using index i $(1 \leq i \leq m)$.

In this paper, feature function f_i is defined as follows.

$$f_i(x, y) = \begin{cases} 1 & if\ \ \tilde{x}_i \in x\ \ and\ \ y = \tilde{y}_i \\ 0 & otherwise \end{cases}$$

We use all combinations of input symbols in x and class labels for features (or the feature function) of MEMs.

With Lagrangian $\lambda = \lambda_1, ..., \lambda_m$, the dual function of H is:

$$\Psi(\lambda) = -\sum_x \tilde{p}(x) \log Z_\lambda(x) + \sum \lambda_i \tilde{p}(f_i),$$

where $Z_\lambda(x) = \sum_y \exp(\sum_i \lambda_i f_i(x, y))$ and $\tilde{p}(x)$ and $\tilde{p}(f_i)$ indicate the empirical distribution of x and f_i in the training data.

The dual optimization problem $\lambda* = \underset{\lambda}{\operatorname{argmax}} \Psi(\lambda)$ can be efficiently solved as an optimization problem without constraints. As a result, probabilistic model $p* = p_\lambda*$ is obtained as:

$$p_{\lambda*}(y|x) = \frac{1}{Z_\lambda(x)} \exp\left(\sum_i \lambda_i f_i(x, y)\right).$$

2.2.2 Applying MEMs to QA

Question analysis is a classification problem that classifies questions into different question types. Answer candidate extraction is also a classification problem that classifies words into answer types (i.e., question types), such as PERSON, DATE, and AWARD. Answer selection is an exactly classification that classifies answer candidates as positive or negative. Therefore, we can apply machine learning techniques to generate classifiers that work as components of a QA system.

In the QBTE approach, these three components, i.e., question analysis, answer candidate extraction, and answer selection, are integrated into one classifier.

To successfully carry out this goal, we have to extract features that reflect properties of correct answers of a question in the context of articles.

3 QBTE Model 1

This section presents a framework, QBTE Model 1, to construct a QA system from question-answer pairs based on the QBTE Approach. When a user gives a question, the framework finds answers to the question in the following two steps.

Document Retrieval retrieves the top N articles or paragraphs from a large-scale corpus.

QBTE creates input data by combining the question features and documents features, evaluates the input data, and outputs the top M answers.[3]

Since this paper focuses on QBTE, this paper uses a simple *idf* method in document retrieval.

Let w_i be words and w_1,w_2,...w_m be a document. Question Answering in the QBTE Model 1 involves directly classifying words w_i in the document into answer words or non-answer words. That is, given input $x^{(i)}$ for w_i, its class label is selected from among $\{I, O, B\}$ as follows:

I: if the word is in the middle of the answer word sequence;

O: if the word is not in the answer word sequence;

B: if the word is the start word of the answer word sequence.

The class labeling system in our experiment is IOB2 (Sang, 2000), which is a variation of IOB (Ramshaw and Marcus, 1995).

Input $x^{(i)}$ of each word is defined as described below.

3.1 Feature Extraction

This paper employs three groups of features as features of input data:

- Question Feature Set (QF);
- Document Feature Set (DF);
- Combined Feature Set (CF), i.e., combinations of question and document features.

[3]In this paper, M is set to 5.

3.1.1 Question Feature Set (QF)

A Question Feature Set (QF) is a set of features extracted only from a question sentence. This feature set is defined as belonging to a question sentence.

The following are elements of a Question Feature Set:

qw: an enumeration of the word n-grams ($1 \leq n \leq N$), e.g., given question "What is CNN?", the features are $\{$*qw:What, qw:is, qw:CNN, qw:What-is, qw:is-CNN* $\}$ if $N = 2$,

qq: interrogative words (e.g., who, where, what, how many),

qm1: POS1 of words in the question, e.g., given "What is CNN?", $\{$ *qm1:wh-adv, qm1:verb, qm1:noun* $\}$ are features,

qm2: POS2 of words in the question,

qm3: POS3 of words in the question,

qm4: POS4 of words in the question.

POS1-POS4 indicate part-of-speech (POS) of the IPA POS tag set generated by the Japanese morphological analyzer ChaSen. For example, "Tokyo" is analyzed as POS1 = *noun*, POS2 = *propernoun*, POS3 = *location*, and POS4 = *general*. This paper used up to 4-grams for qw.

3.1.2 Document Feature Set (DF)

Document Feature Set (DF) is a feature set extracted only from a document. Using only DF corresponds to *unbiased* Term Extraction (TE).

For each word w_i, the following features are extracted:

dw–k,. . .,dw+0,. . .,dw+k: k preceding and following words of the word w_i, e.g., $\{$ *dw–1*:w_{i-1}, *dw+0*:w_i, *dw+1*:$w_{i+1}\}$ if $k = 1$,

dm1–k,. . .,dm1+0,. . .,dm1+k: POS1 of k preceding and following words of the word w_i,

dm2–k,. . .,dm2+0,. . .,dm2+k: POS2 of k preceding and following words of the word w_i,

dm3–k,. . .,dm3+0,. . .,dm3+k: POS3 of k preceding and following words of the word w_i,

dm4–k,. . .,dm4+0,. . .,dm4+k: POS4 of k preceding and following words of the word w_i.

In this paper, k is set to 3 so that the window size is 7.

3.1.3 Combined Feature Set (CF)

Combined Feature Set (CF) contains features created by combining question features and document features. QBTE Model 1 employs CF. For each word w_i, the following features are created.

cw–k,. . .,cw+0,. . .,cw+k: matching results (true/false) between each of dw–k,...,dw+k features and any qw feature, e.g., *cw–1:true* if *dw–1:President* and *qw: President*,

cm1–k,. . .,cm1+0,. . .,cm1+k: matching results (true/false) between each of dm1–k,...,dm1+k features and any POS1 in qm1 features,

cm2–k,. . .,cm2+0,. . .,cm2+k: matching results (true/false) between each of dm2–k,...,dm2+k features and any POS2 in qm2 features,

cm3–k,. . .,cm3+0,. . .,cm3+k: matching results (true/false) between each of dm3–k,...,dm3+k features and any POS3 in qm3 features,

cm4–k,. . .,cm4+0,. . .,cm4+k: matching results (true/false) between each of dm4–k,...,dm4+k features and any POS4 in qm4 features,

cq–k,. . .,cq+0,. . .,cq+k: combinations of each of dw–k,...,dw+k features and qw features, e.g., *cq–1:President&Who* is a combination of *dw–1:President* and *qw:Who*.

3.2 Training and Execution

The training phase estimates a probabilistic model from training data ($x^{(1)}$,$y^{(1)}$),...,($x^{(n)}$,$y^{(n)}$) generated from the CRL QA Data. The execution phase evaluates the probability of $y'^{(i)}$ given input$x'^{(i)}$ using the the probabilistic model.

Training Phase

1. Given question q, correct answer a, and document d.
2. Annotate $\langle A\rangle$ and $\langle/A\rangle$ right before and after answer a in d.
3. Morphologically analyze d.
4. For $d = w_1, ..., \langle A\rangle, w_j, ..., w_k, \langle/A\rangle, ..., w_m$, extract features as $x^{(1)}$,...,$x^{(m)}$.
5. Class label $y^{(i)} = B$ if w_i follows $\langle A\rangle$, $y^{(i)} = I$ if w_i is inside of $\langle A\rangle$ and $\langle/A\rangle$, and $y^{(i)} = O$ otherwise.

Table 2: Main Results with 10-fold Cross Validation

	Correct Answer Rank					MRR	Top5
	1	2	3	4	5		
Exact match	453	139	68	35	19	0.28	0.36
Partial match	684	222	126	80	48	0.43	0.58
Ave.						0.355	0.47
Manual evaluation	578	188	86	55	34	0.36	0.47

6. Estimate $p_\lambda*$ from ($x^{(1)}$,$y^{(1)}$),...,($x^{(n)}$,$y^{(n)}$) using Maximum Entropy Models.

The execution phase extracts answers from retrieved documents as Term Extraction, biased by the question.

Execution Phase

1. Given question q and paragraph d.
2. Morphologically analyze d.
3. For w_i of $d = w_1, ..., w_m$, create input data $x'^{(i)}$ by extracting features.
4. For each $y'^{(j)} \in \mathcal{Y}$, compute $p_\lambda * (y'^{(j)}|x'^{(i)})$, which is a probability of $y'^{(j)}$ given $x'^{(i)}$.
5. For each $x'^{(i)}$, $y'^{(j)}$ with the highest probability is selected as the label of w_i.
6. Extract word sequences that start with the word labeled B and are followed by words labeled I from the labeled word sequence of d.
7. Rank the top M answers according to the probability of the first word.

This approach is designed to extract only the most highly probable answers. However, pin-pointing only answers is not an easy task. To select the top five answers, it is necessary to loosen the condition for extracting answers. Therefore, in the execution phase, we only give label O to a word if its probability exceeds 99%, otherwise we give the second most probable label.

As a further relaxation, word sequences that include B inside the sequences are extracted for answers. This is because our preliminary experiments indicated that it is very rare for two answer candidates to be adjacent in Question-Biased Term Extraction, unlike an ordinary Term Extraction task.

4 Experimental Results

We conducted 10-fold cross validation using the CRL QA Data. The output is evaluated using the Top5 score and MRR.

Top5 Score shows the rate at which at least one correct answer is included in the top 5 answers.

MRR (Mean Reciprocal Rank) is the average reciprocal rank ($1/n$) of the highest rank n of a correct answer for each question.

Judgment of whether an answer is correct is done by both automatic and manual evaluation. Automatic evaluation consists of exact matching and partial matching. Partial matching is useful for absorbing the variation in extraction range. A partial match is judged correct if a system's answer completely includes the correct answer or the correct answer completely includes a system's answer. Table 2 presents the experimental results. The results show that a QA system can be built by using our QBTE approach. The manually evaluated performance scored MRR=0.36 and Top5=0.47. However, manual evaluation is costly and time-consuming, so we use automatic evaluation results, i.e., exact matching results and partial matching results, as a pseudo lower-bound and upper-bound of the performances. Interestingly, the manual evaluation results of MRR and Top5 are nearly equal to the average between exact and partial evaluation.

To confirm that the QBTE ranks potential answers to the higher rank, we changed the number of paragraphs retrieved from a large corpus from $N = 1, 3, 5$ to 10. Table 3 shows the results. Whereas the performances of Term Extraction (TE) and Term Extraction with question features (TE+QF) significantly degraded, the performance of the QBTE (CF) did not severely degrade with the larger number of retrieved paragraphs.

Table 3: Answer Extraction from Top N documents

Feature set	Top N paragraphs	Match	Correct Answer Rank					MRR	Top5
			1	2	3	4	5		
TE (DF)	1	Exact	102	109	80	71	62	**0.11**	**0.21**
		Partial	207	186	155	153	121	**0.21**	**0.41**
	3	Exact	65	63	55	53	43	0.07	0.14
		Partial	120	131	112	108	94	0.13	0.28
	5	Exact	51	38	38	36	36	0.05	0.10
		Partial	99	80	89	81	75	0.10	0.21
	10	Exact	29	17	19	22	18	0.03	0.07
		Partial	59	38	35	49	46	0.07	0.14
TE (DF) + QF	1	Exact	120	105	94	63	80	**0.12**	**0. 23**
		Partial	207	198	175	126	140	**0.21**	**0 .42**
	3	Exact	65	68	52	58	57	0.07	0.15
		Partial	119	117	111	122	106	0.13	0.29
	5	Exact	44	57	41	35	31	0.05	0.10
		Partial	91	104	71	82	63	0.10	0.21
	10	Exact	28	42	30	28	26	0.04	0.08
		Partial	57	68	57	56	45	0.07	0.14
QBTE (CF)	1	Exact	453	139	68	35	19	**0.28**	0.36
		Partial	684	222	126	80	48	**0.43**	0.58
	3	Exact	403	156	92	52	43	0.27	**0.37**
		Partial	539	296	145	105	92	0.42	**0.62**
	5	Exact	381	153	92	59	50	0.26	**0.37**
		Partial	542	291	164	122	102	0.40	0.61
	10	Exact	348	128	92	65	57	0.24	0.35
		Partial	481	257	173	124	102	0.36	0.57

5 Discussion

Our approach needs no question type system, and it still achieved 0.36 in MRR and 0.47 in Top5. This performance is comparable to the results of SAIQA-II (Sasaki et al., 2004) (MRR=0.4, Top5=0.55) whose question analysis, answer candidate extraction, and answer selection modules were independently built from a QA dataset and an NE dataset, which is limited to eight named entities, such as PERSON and LOCATION. Since the QA dataset is not publicly available, it is not possible to directly compare the experimental results; however we believe that the performance of the QBTE Model 1 is comparable to that of the conventional approaches, even though it does not depend on question types, named entities, or class names.

Most of the partial answers were judged correct in manual evaluation. For example, for "How many times bigger ...?", "two times" is a correct answer but "two" was judged correct. Suppose that "John Kerry" is a prepared correct answer in the CRL QA Data. In this case, "Senator John Kerry" would also be correct. Such additions and omissions occur because our approach is not restricted to particular extraction units, such as named entities or class names.

The performance of QBTE was affected little by the larger number of retrieved paragraphs, whereas the performances of TE and TE + QF significantly degraded. This indicates that QBTE Model 1 is not mere Term Extraction with document retrieval but Term Extraction appropriately biased by questions.

Our experiments used no information about question types given in the CRL QA Data because we are seeking a universal method that can be used for any QA dataset. *Beyond this main goal*, as a reference, The Appendix shows our experimental results classified into question types without using them in the training phase. The results of automatic evaluation of complete matching are in Top5 (T5), and MRR and partial matching are in Top5 (T5') and MRR'. It is interesting that minor question types were correctly answered, e.g., SEA and WEAPON, for which there was only one training question.

We also conducted an additional experiment, as a reference, on the training data that included question types defined in the CRL QA Data; the question-type of each question is added to the qw feature. The performance of QBTE from the first-ranked paragraph showed no difference from that of experiments shown in Table 2.

6 Related Work

There are two previous studies on integrating QA components into one using machine learning/statistical NLP techniques. Echihabi et al. (Echihabi et al., 2003) used Noisy-Channel Models to construct a QA system. In this approach, the range of Term Extraction is not trained by a data set but selected from answer candidates, e.g., named entities and noun phrases, generated by a decoder. Lita et al. (Lita and Carbonell, 2004) share our motivation to build a QA system only from question-answer pairs without depending on the question types. Their method finds clusters of questions and defines how to answer questions in each cluster. However, their approach is to find snippets, i.e., short passages including answers, not exact answers extracted by Term Extraction.

7 Conclusion

This paper described a novel approach to extracting answers to a question using probabilistic models constructed from only question-answer pairs. This approach requires no question type system, no named entity extractor, and no class name extractor. To the best of our knowledge, no previous study has regarded Question Answering as Question-Biased Term Extraction. As a feasibility study, we built a QA system using Maximum Entropy Models on a 2000-question/answer dataset. The results were evaluated by 10-fold cross validation, which showed that the performance is 0.36 in MRR and 0.47 in Top5. Since this approach relies on a morphological analyzer, applying the QBTE Model 1 to QA tasks of other languages is our future work.

Acknowledgment

This research was supported by a contract with the National Institute of Information and Communications Technology (NICT) of Japan entitled, "A study of speech dialogue translation technology based on a large corpus".

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra: A Maximum Entropy Approach to Natural Language Processing, *Computational Linguistics*, Vol. 22, No. 1, pp. 39–71 (1996).

Abdessamad Echihabi and Daniel Marcu: A Noisy-Channel Approach to Question Answering, *Proc. of ACL-2003*, pp. 16-23 (2003).

Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi: Question Answering Using Maximum-Entropy Components, *Proc. of NAACL-2001* (2001).

Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi: IBM's Statistical Question Answering System – TREC-10, *Proc. of TREC-10* (2001).

Lucian Vlad Lita and Jaime Carbonell: Instance-Based Question Answering: A Data-Driven Approach: *Proc. of EMNLP-2004*, pp. 396–403 (2004).

Hwee T. Ng, Jennifer L. P. Kwan, and Yiyuan Xia: Question Answering Using a Large Text Database: A Machine Learning Approach: *Proc. of EMNLP-2001*, pp. 67–73 (2001).

Marisu A. Pasca and Sanda M. Harabagiu: High Performance Question/Answering, *Proc. of SIGIR-2001*, pp. 366–374 (2001).

Lance A. Ramshaw and Mitchell P. Marcus: Text Chunking using Transformation-Based Learning, *Proc. of WVLC-95*, pp. 82–94 (1995).

Erik F. Tjong Kim Sang: Noun Phrase Recognition by System Combination, *Proc. of NAACL-2000*, pp. 55–55 (2000).

Yutaka Sasaki, Hideki Isozaki, Jun Suzuki, Kouji Kokuryou, Tsutomu Hirao, Hideto Kazawa, and Eisaku Maeda, SAIQA-II: A Trainable Japanese QA System with SVM, *IPSJ Journal*, Vol. 45, NO. 2, pp. 635-646, 2004. (in Japanese)

Satoshi Sekine, Kiyoshi Sudo, Yusuke Shinyama, Chikashi Nobata, Kiyotaka Uchimoto, and Hitoshi Isahara, NYU/CRL QA system, QAC question analysis and CRL QA data, *in Working Notes of NTCIR Workshop 3* (2002).

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

Jun Suzuki, Hirotoshi Taira, Yutaka Sasaki, and Eisaku Maeda: Directed Acyclic Graph Kernel, *Proc. of ACL 2003 Workshop on Multilingual Summarization and Question Answering - Machine Learning and Beyond*, pp. 61–68, Sapporo (2003).

Ingrid Zukerman and Eric Horvitz: Using Machine Learning Techniques to Interpret WH-Questions, *Proc. of ACL-2001*, Toulouse, France, pp. 547–554 (2001).

Appendix: Analysis of Evaluation Results w.r.t. Question Type — Results of QBTE from the first-ranked paragraph (NB: No information about these question types was used in the training phrase.)

Question Type	#Qs	MRR	T5	MRR'	T5'
GOE	36	0.30	0.36	0.41	0.53
GPE	4	0.50	0.50	1.00	1.00
N_EVENT	7	0.76	0.86	0.76	0.86
EVENT	19	0.17	0.21	0.41	0.53
GROUP	74	0.28	0.35	0.45	0.62
SPORTS_TEAM	15	0.28	0.40	0.45	0.73
BROADCAST	1	0.00	0.00	0.00	0.00
POINT	2	0.00	0.00	0.00	0.00
DRUG	2	0.00	0.00	0.00	0.00
SPACESHIP	4	0.88	1.00	0.88	1.00
ACTION	18	0.22	0.22	0.30	0.44
MOVIE	6	0.50	0.50	0.56	0.67
MUSIC	8	0.19	0.25	0.56	0.62
WATER_FORM	3	0.50	0.67	0.50	0.67
CONFERENCE	17	0.14	0.24	0.46	0.65
SEA	1	1.00	1.00	1.00	1.00
PICTURE	1	0.00	0.00	0.00	0.00
SCHOOL	21	0.10	0.10	0.33	0.43
ACADEMIC	5	0.20	0.20	0.37	0.60
PERCENT	47	0.35	0.43	0.43	0.55
COMPANY	77	0.45	0.55	0.57	0.70
PERIODX	1	1.00	1.00	1.00	1.00
RULE	35	0.30	0.43	0.49	0.69
MONUMENT	2	0.00	0.00	0.25	0.50
SPORTS	9	0.17	0.22	0.40	0.67
INSTITUTE	26	0.38	0.46	0.53	0.69
MONEY	110	0.33	0.40	0.48	0.63
AIRPORT	4	0.38	0.50	0.44	0.75
MILITARY	4	0.00	0.00	0.25	0.25
ART	4	0.25	0.50	0.25	0.50
MONTH_PERIOD	6	0.06	0.17	0.06	0.17
LANGUAGE	3	1.00	1.00	1.00	1.00
COUNTX	10	0.33	0.40	0.38	0.60
AMUSEMENT	2	0.00	0.00	0.00	0.00
PARK	1	0.00	0.00	0.00	0.00
SHOW	3	0.78	1.00	1.11	1.33
PUBLIC_INST	19	0.18	0.26	0.34	0.53
PORT	3	0.17	0.33	0.33	0.67
N_COUNTRY	8	0.28	0.38	0.32	0.50
NATIONALITY	4	0.50	0.50	1.00	1.00
COUNTRY	84	0.45	0.60	0.51	0.67
OFFENSE	9	0.23	0.44	0.23	0.44
CITY	72	0.41	0.50	0.53	0.65
N_FACILITY	4	0.25	0.25	0.38	0.50
FACILITY	11	0.20	0.36	0.25	0.55
TIMEX	3	0.00	0.00	0.00	0.00
TIME_TOP	2	0.00	0.00	0.50	0.50
TIME_PERIOD	8	0.12	0.12	0.48	0.75
TIME	13	0.22	0.31	0.29	0.38
ERA	3	0.00	0.00	0.33	0.33
PHENOMENA	5	0.50	0.60	0.60	0.80
DISASTER	4	0.50	0.75	0.50	0.75
OBJECT	5	0.47	0.60	0.47	0.60
CAR	1	1.00	1.00	1.00	1.00
RELIGION	5	0.30	0.40	0.30	0.40
WEEK_PERIOD	4	0.05	0.25	0.55	0.75
WEIGHT	12	0.21	0.25	0.31	0.42
PRINTING	6	0.17	0.17	0.38	0.50

Question Type	#Q	MRR	T5	MRR'	T5'
RANK	7	0.18	0.29	0.54	0.71
BOOK	6	0.31	0.50	0.47	0.67
AWARD	9	0.17	0.33	0.34	0.56
N_LOCATION	2	0.10	0.50	0.10	0.50
VEGETABLE	10	0.31	0.50	0.34	0.60
COLOR	5	0.20	0.20	0.20	0.20
NEWSPAPER	7	0.61	0.71	0.61	0.71
WORSHIP	8	0.47	0.62	0.62	0.88
SEISMIC	1	0.00	0.00	1.00	1.00
N_PERSON	72	0.30	0.39	0.43	0.60
PERSON	282	0.18	0.21	0.46	0.55
NUMEX	19	0.32	0.32	0.35	0.47
MEASUREMENT	1	0.00	0.00	0.00	0.00
P_ORGANIZATION	3	0.33	0.33	0.67	0.67
P_PARTY	37	0.30	0.41	0.43	0.57
GOVERNMENT	37	0.50	0.54	0.53	0.57
N_PRODUCT	41	0.25	0.37	0.37	0.56
PRODUCT	58	0.24	0.34	0.44	0.69
WAR	2	0.75	1.00	0.75	1.00
SHIP	7	0.26	0.43	0.40	0.57
N_ORGANIZATION	20	0.14	0.25	0.28	0.55
ORGANIZATION	23	0.08	0.13	0.20	0.30
SPEED	1	0.00	0.00	1.00	1.00
VOLUME	5	0.00	0.00	0.18	0.60
GAMES	8	0.28	0.38	0.34	0.50
POSITION_TITLE	39	0.20	0.28	0.30	0.44
REGION	22	0.17	0.23	0.46	0.64
GEOLOGICAL	3	0.42	0.67	0.42	0.67
LOCATION	2	0.00	0.00	0.50	0.50
EXTENT	22	0.04	0.09	0.13	0.18
CURRENCY	1	0.00	0.00	0.00	0.00
STATION	3	0.50	0.67	0.50	0.67
RAILROAD	1	0.00	0.00	0.25	1.00
PHONE	1	0.00	0.00	0.00	0.00
PROVINCE	36	0.30	0.33	0.45	0.50
N_ANIMAL	3	0.11	0.33	0.22	0.67
ANIMAL	10	0.26	0.50	0.31	0.60
ROAD	1	0.00	0.00	0.50	1.00
DATE_PERIOD	9	0.11	0.11	0.33	0.33
DATE	130	0.24	0.32	0.41	0.58
YEAR_PERIOD	34	0.22	0.29	0.38	0.59
AGE	22	0.34	0.45	0.44	0.59
MULTIPLICATION	9	0.39	0.44	0.56	0.67
CRIME	4	0.75	0.75	0.75	0.75
AIRCRAFT	2	0.00	0.00	0.25	0.50
MUSEUM	3	0.33	0.33	0.33	0.33
DISEASE	18	0.29	0.50	0.43	0.72
FREQUENCY	13	0.18	0.31	0.19	0.38
WEAPON	1	1.00	1.00	1.00	1.00
MINERAL	18	0.16	0.22	0.25	0.39
METHOD	29	0.39	0.48	0.48	0.62
ETHNIC	3	0.42	0.67	0.75	1.00
NAME	5	0.20	0.20	0.40	0.40
SPACE	4	0.50	0.50	0.50	0.50
THEORY	1	0.00	0.00	0.00	0.00
LANDFORM	5	0.13	0.40	0.13	0.40
TRAIN	2	0.17	0.50	0.17	0.50
	2000	0.28	0.36	0.43	0.58

Exploring and Exploiting the Limited Utility of Captions in Recognizing Intention in Information Graphics[*]

Stephanie Elzer[1] and **Sandra Carberry**[2] and **Daniel Chester**[2] and **Seniz Demir**[2] and **Nancy Green**[3] and **Ingrid Zukerman**[4] and **Keith Trnka**[2]

[1]Dept. of Computer Science, Millersville University, Millersville, PA 17551
[2]Dept. of Computer Science, University of Delaware, Newark, DE 19716
[3]Dept. of Mathematical Sciences, Univ. of NC at Greensboro, Greensboro, NC 27402
[4]School of CS & Software Engrg, Monash Univ., Clayton, Victoria 3800 Australia

Abstract

This paper presents a corpus study that explores the extent to which captions contribute to recognizing the intended message of an information graphic. It then presents an implemented graphic interpretation system that takes into account a variety of communicative signals, and an evaluation study showing that evidence obtained from shallow processing of the graphic's caption has a significant impact on the system's success. This work is part of a larger project whose goal is to provide sight-impaired users with effective access to information graphics.

1 Introduction

Language research has posited that a speaker or writer executes a speech act whose intended meaning he expects the listener to be able to deduce, and that the listener identifies the intended meaning by reasoning about the observed signals and the mutual beliefs of author and interpreter (Grice, 1969; Clark, 1996). But as noted by Clark (Clark, 1996), language is more than just words. It is any "signal" (or lack of signal when one is expected), where a signal is a deliberate action that is intended to convey a message.

Although some information graphics are only intended to display data values, the overwhelming majority of the graphics that we have examined (taken from newspaper, magazine, and web articles) appear to have some underlying goal or intended message, such as the graphic in Figure 1 whose communicative goal is ostensibly to convey the sharp increase in local bankruptcies in the current year compared with the previous decreasing trend. Applying Clark's view of language, it is reasonable to presume that the author of an information graphic expects the viewer to deduce from the graphic the message that the graphic was intended to convey, by reasoning about the graphic itself, the salience of entities in the graphic, and the graphic's caption.

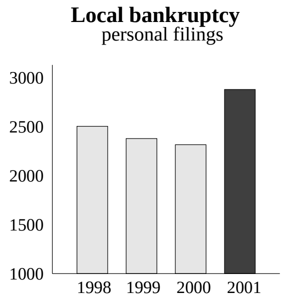


Figure 1: Graphic from a 2001 Local Newspaper

This paper adopts Clark's view of language as any deliberate signal that is intended to convey a message. Section 3 investigates the kinds of signals used in information graphics. Section 4 presents a corpus study that investigates the extent to which captions capture the message of the graphic, illustrates the issues that would arise in trying to fully understand such captions, and proposes shallow processing of the caption to extract evidence from it. Section 5 then describes how evidence obtained from a variety of communicative signals, including shallow processing of the graphic's caption, is used in a probabilistic system for hypothesizing the intended message of the graphic. Section 6 presents an eval-

[*]Authors can be reached via email as follows: elzer@cs.millersville.edu, nlgreen@uncg.edu, {carberry, chester, demir, trnka}@cis.udel.edu, Ingrid.Zukerman@infotech.monash.edu.au.

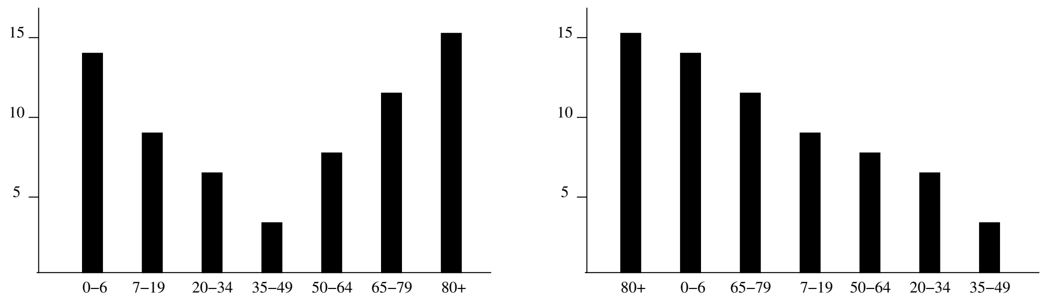


Figure 2: Two Alternative Graphs from the Same Data

uation showing the system's success, with particular attention given to the impact of evidence from shallow processing of the caption, and Section 7 discusses future work.

Although we believe that our findings are extendible to other kinds of information graphics, our current work focuses on bar charts. This research is part of a larger project whose goal is a natural language system that will provide effective access to information graphics for individuals with sight impairments, by inferring the intended message underlying the graphic, providing an initial summary of the graphic that includes the intended message along with notable features of the graphic, and then responding to follow-up questions from the user.

2 Related Work

Our work is related to efforts on graph summarization. (Yu et al., 2002) used pattern recognition techniques to summarize interesting features of automatically generated graphs of time-series data from a gas turbine engine. (Futrelle and Nikolakis, 1995) developed a constraint grammar for parsing vector-based visual displays and producing representations of the elements comprising the display. The goal of Futrelle's project is to produce a graphic that summarizes one or more graphics from a document (Futrelle, 1999). The summary graphic might be a simplification of a graphic or a merger of several graphics from the document, along with an appropriate summary caption. Thus the end result of summarization will itself be a graphic. The long range goal of our project, on the other hand, is to provide alternative access to information graphics via an initial textual summary followed by an interactive follow-up component for additional information. The intended message of the graphic will be an important component of the initial summary, and hypothesizing it is the goal of our current work.

3 Evidence about the Intended Message

The graphic designer has many alternative ways of designing a graphic; different designs contain different communicative signals and thus convey different communicative intents. For example, consider the two graphics in Figure 2. The graphic in Figure 2a conveys that average doctor visits per year is U-shaped by age; it starts out high when one is very young, decreases into middle age, and then rises again as one ages. The graphic in Figure 2b presents the same data; but instead of conveying a trend, this graphic seems to convey that the elderly and the young have the highest number of doctor visits per year. These graphics illustrate how choice of design affects the message that the graphic conveys.

Following the AutoBrief work (Kerpedjiev and Roth, 2000) (Green *et al.*, 2004) on generating graphics that fulfill communicative goals, we hypothesize that the designer chooses a design that best facilitates the perceptual and cognitive tasks that are most important to conveying his intended message, subject to the constraints imposed by competing tasks. By *perceptual tasks* we mean tasks that can be performed by simply viewing the graphic, such as finding the top of a bar in a bar chart; by *cognitive tasks* we mean tasks that are done via mental computations, such as computing the difference between two numbers.

Thus one source of evidence about the intended message is the relative difficulty of the perceptual tasks that the viewer would need to perform in order to recognize the message. For example, determining the entity with maximum value in a bar chart will be easiest if the bars are arranged in ascending or descending order of height. We have constructed a set of rules, based on research by cognitive psychologists, that estimate the relative difficulty of performing different perceptual tasks; these rules have been validated by eye-tracking experiments and are presented in (Elzer *et al.*, 2004).

Another source of evidence is entities that have been made salient in the graphic by some kind of focusing device, such as coloring some elements of the graphic, annotations such as an asterisk, or an arrow pointing to a particular location in a graphic. Entities that have been made salient suggest particular instantiations of perceptual tasks that the viewer is expected to perform, such as comparing the heights of two highlighted bars in a bar chart.

And lastly, one would expect captions to help convey the intended message of an information graphic. The next section describes a corpus study that we performed in order to explore the usefulness of captions and how we might exploit evidence from them.

4 A Corpus Study of Captions

Although one might suggest relying almost exclusively on captions to interpret an information graphic, (Corio and Lapalme, 1999) found in a corpus study that captions are often very general. The objective of their corpus study was to categorize the kinds of information in captions so that their findings could be used in forming rules for generating graphics with captions.

Our project is instead concerned with recognizing the intended message of an information graphic. To investigate how captions might be used in a system for understanding information graphics, we performed a corpus study in which we analyzed the first 100 bar charts from our corpus of information graphics; this corpus contains a variety of bar charts from different publication venues. The following subsections present the results of this corpus study.

4.1 Do Captions Convey the Intended Message?

Our first investigation explored the extent to which captions capture the intended message of an information graphic. We extracted the first 100 graphics from our corpus of bar charts. The intended message of each bar chart had been previously annotated by two coders. The coders were asked to identify 1) the intended message of the graphic using a list of 12 high-level intentions (see Section 5 for examples) and 2) the instantiation of the parameters. For example, if the coder classified the intended message of a graphic as *Change-trend*, the coder was also asked to identify where the first trend began, its general slope (increasing, decreasing, or stable), where the change in trend occurred, the end of the second trend, and the slope of the second trend. If there was disagreement between the coders on either the intention or the instantiation of the parameters, we utilized consensus-based annotation (Ang *et al.*, 2002), in which the coders discussed the graphic to try to come to an agreement. As observed by (Ang *et al.*, 2002), this allowed us to include the "harder" or less obvious graphics in our study, thus lowering our expected system performance. We then examined the caption of each graphic, and determined to what extent the caption captured the graphic's intended message. Figure 3 shows the results. 44% of the captions in our corpus did not convey to any extent the message of the information graphic. The following categorizes the purposes that these captions served, along with an example of each:

Category	**#**
Category-1: Captures intention (mostly)	34
Category-2: Captures intention (somewhat)	15
Category-3: Hints at intention	7
Category-4: No contribution to intention	44

Figure 3: Analysis of 100 Captions on Bar Charts

- general heading (8 captions): *"UGI Monthly Gas Rates"* on a graphic conveying a recent spike in home heating bills.

- reference to dependent axis (15 captions): *"Lancaster rainfall totals for July"* on a graphic conveying that July-02 was the driest of the previous decade.

- commentary relevant to graphic (4 captions): *"Basic performers: One look at the best performing stocks in the Standard&Poor's 500 index this year shows that companies with basic businesses are rewarding investors"* on a graphic conveying the relative rank of different stocks, some of which were basic businesses and some of which were not. This type of information was classified as *deductive* by (Corio and Lapalme, 1999) since it draws a conclusion from the data depicted in the graphic.

- commentary extending message of graphic (8 captions): *"Profits are getting squeezed"* on a graphic conveying that Southwest Airlines net income is estimated to increase in 2003 after falling the preceding three years. Here the commentary does not draw a conclusion from the data in the graphic but instead supplements the graphic's message. However this type of caption would probably fall into the *deductive* class in (Corio and Lapalme, 1999).

- humor (7 captions): *"The Sound of Sales"* on a graphic conveying the changing trend (downward after years of increase) in record album sales. This caption has nothing to do with the change-trend message of the graphic, but appears to be an attempt at humor.

- conclusion unwarranted by graphic (2 captions): *"Defense spending declines"* on a graphic that in fact conveys that recent defense spending is increasing.

Slightly over half the captions (56%) contributed to understanding the graphic's intended message. 34% were judged to convey most of the intended message. For example, the caption *"Tennis players top nominees"* appeared on a graphic whose intended message is to convey that more tennis players were nominated for the 2003 Laureus World Sports Award than athletes from any other sport. Since we argue that captions alone are insufficient for interpreting information graphics, in the few cases where it was unclear whether a caption should be placed in Category-1 or Category-2, we erred on the side of over-rating the contribution of a caption to the graphic's intended message. For example, consider the caption *"Chirac is riding high in the polls"* which appeared on a graphic conveying that there has been a steady increase in Chirac's approval ratings from 55% to about 75%. Although this caption does not fully capture the communicative intention of the graphic (since it does not capture the steady increase conveyed by the graphic), we placed it in the first category since one might argue that *riding high in the polls* would suggest both high and improving ratings.

15% of the captions were judged to convey only part of the graphic's intended message; an example is *"Drug spending for young outpace seniors"* that appears on a graphic whose intended message appears to be that there is a downward trend by age for increased drug spending; we classified the caption in Category-2 since the caption fails to capture that the graphic is talking about percent increases in drug spending, not absolute drug spending, and that the graphic conveys the downward trend for increases in drug spending by age group, not just that increases for the young were greater than for the elderly.

7% of the captions were judged to only hint at the graphic's message. An example is *"GM's Money Machine"* which appeared on a graphic whose intended message was a contrast of recent performance against the previous trend — ie., that although there had been a steady decrease in the percentage of GM's overall income produced by its finance unit, there was now a substantial increase in the percentage provided by the finance unit. Since the term *money machine* is a colloquialism that suggests making a lot of money, the caption was judged to *hint at* the graphic's intended message.

4.2 Understanding Captions

For the 49 captions in Category 1 or 2 (where the caption conveyed at least some of the message of the graphic), we examined how well the caption could be parsed and understood by a natural language system. We found that 47% were fragments (for example, *"A Growing Biotech Market"*), or involved some other kind of ill-formedness (for example, *"Running tops in sneaker wear in 2002"* or *"More seek financial aid"*[1]). 16% would require extensive domain knowledge or analogical reasoning to understand. One example is *"Chirac is riding high in the polls"* which would require understanding the meaning of *riding high in the polls*. Another example is *"Bad Moon Rising"*; here the verb *rising* suggests that something is increasing, but the

[1]Here we judge the caption to be ill-formed due to the ellipsis since *More* should be *More students*.

system would need to understand that a *bad moon* refers to something undesirable (in this case, delinquent loans).

4.3 Simple Evidence from Captions

Although our corpus analysis showed that captions can be helpful in understanding the message conveyed by an information graphic, it also showed that full understanding of a caption would be problematic; moreover, once the caption was understood, we would still need to relate it to the information extracted from the graphic itself, which appears to be a difficult problem.

Thus we began investigating whether shallow processing of the caption might provide evidence that could be effectively combined with other evidence obtained from the graphic itself. Our analysis provided the following observations:

- Verbs in a caption often suggest the kind of message being conveyed by the graphic. An example from our corpus is *"Boating deaths decline"*; the verb *decline* suggests that the graphic conveys a decreasing trend. Another example from our corpus is *"American Express total billings still lag"*; the verb *lag* suggests that the graphic conveys that some entity (in this case *American Express*) is ranked behind some others.

- Adjectives in a caption also often suggest the kind of message being conveyed by the graphic. An example from our corpus is *"Air Force has largest percentage of women"*; the adjective *largest* suggests that the graphic is conveying an entity whose value is largest. Adjectives derived from verbs function similarly to verbs. An example from our corpus is *"Soaring Demand for Servers"* which is the caption on a graphic that conveys the rapid increase in demand for servers. Here the adjective *soaring* is derived from the verb *soar*, and suggests that the graphic is conveying a strong increase.

- Nouns in a caption often refer to an entity that is a label on the independent axis. When this occurs, the caption brings the entity into focus and suggests that it is part of the intended message of the graphic. An example from our corpus is *"Germans miss their marks"* where the graphic displays a bar chart that is intended to convey that Germans are the least happy with the Euro. Words that usually appear as verbs, but are used in the caption as a noun, may function similarly to verbs. An example is *"Cable On The Rise"*; in this caption, *rise* is used as a noun, but suggests that the graphic is conveying an increase.

5 Utilizing Evidence

We developed and implemented a probabilistic framework for utilizing evidence from a graphic and its caption to hypothesize the graphic's intended message. To identify the intended message of a new information graphic, the graphic is first given to a Visual Extraction Module (Chester and Elzer, 2005) that is responsible for recognizing the individual components of a graphic, identifying the relationship of the components to one another and to the graphic as a whole, and classifying the graphic as to type (bar chart, line graph, etc.); the result is an XML file that describes the graphic and all of its components.

Next a Caption Processing Module analyzes the caption. To utilize verb-related evidence from captions, we identified a set of verbs that would indicate each category of high-level goal[2], such as *recover* for Change-trend and *beats* for Relative-difference; we then extended the set of verbs by examining WordNet for verbs that were closely related in meaning, and constructed a verb class for each set of closely related verbs. Adjectives such as *more* and *most* were handled in a similar manner. The Caption Processing Module applies a part-of-speech tagger and a stemmer to the caption in order to identify nouns, adjectives, and the root form of verbs and adjectives derived from verbs. The XML representation of the graphic is augmented to indicate any independent axis labels that match nouns in the caption, and the presence of a verb or adjective class in the caption.

The Intention Recognition Module then analyzes the XML file to build the appropriate Bayesian network; the current system is limited to bar charts, but

[2]As described in the next paragraph, there are 12 categories of high-level goals.

the principles underlying the system should be extendible to other kinds of information graphics. The network is described in (Elzer *et al.*, 2005). Very briefly, our analysis of simple bar charts has shown that the intended message can be classified into one of 12 high-level goals; examples of such goals include:

- Change-trend: *Viewer to believe that there is a <slope-1> trend from <param1> to <param2> and a significantly different <slope-2> trend from <param3> to <param4>*
- Relative-difference: *Viewer to believe that the value of element <param1> is <comparison> the value of element <param2>* where <comparison> is *greater-than*, *less-than*, or *equal-to*.

Each category of high-level goal is represented by a node in the network (whose parent is the top-level goal node), and instances of these goals (ie., goals with their parameters instantiated) appear as children with inhibitory links (Huber et al., 1994) capturing their mutual exclusivity. Each goal is broken down further into subtasks (perceptual or cognitive) that the viewer would need to perform in order to accomplish the goal of the parent node. The network is built dynamically when the system is presented with a new information graphic, so that nodes are added to the network only as suggested by the graphic. For example, low-level nodes are added for the easiest primitive perceptual tasks and for perceptual tasks in which a parameter is instantiated with a salient entity (such as an entity colored differently from others in the graphic or an entity that appears as a noun in the caption), since the graphic designer might have intended the viewer to perform these tasks; then higher-level goals that involve these tasks are added, until eventually a link is established to the top-level goal node.

Next evidence nodes are added to the network to capture the kinds of evidence noted in Sections 3 and 4.3. For example, evidence nodes are added to the network as children of each low-level perceptual task; these evidence nodes capture the relative difficulty (categorized as easy, medium, hard, or impossible) of performing the perceptual task as estimated by our effort estimation rules mentioned in Section 3, whether a parameter in the task refers to an entity that is salient in the graphic, and whether a parameter in the task refers to an entity that is a noun in the caption. An evidence node, indicating for each verb class whether that verb class appears in the caption (either as a verb, or as an adjective derived from a verb, or as a noun that can also serve as a verb) is added as a child of the top level goal node. Adjectives such as *more* and *most* that provide evidence are handled in a similar manner.

In a Bayesian network, conditional probability tables capture the conditional probability of a child node given the value of its parent(s). For example, the network requires the conditional probability of an entity appearing as a noun in the caption given that recognizing the intended message entails performing a particular perceptual task involving that entity. Similarly, the network requires the conditional probability, for each class of verb, that the verb class appears in the caption given that the intended message falls into a particular intention category. These probabilities are learned from our corpus of graphics, as described in (Elzer *et al.*, 2005).

6 Evaluation

In this paper, we are particularly interested in whether shallow processing of captions can contribute to recognizing the intended message of an information graphic. As mentioned earlier, the intended message of each information graphic in our corpus of bar charts had been previously annotated by two coders. To evaluate our approach, we used leave-one-out cross validation. We performed a series of experiments in which each graphic in the corpus is selected once as the test graphic, the probability tables in the Bayesian network are learned from the remaining graphics, and the test graphic is presented to the system as a test case. The system was judged to fail if either its top-rated hypothesis did not match the intended message that was assigned to the graphic by the coders or the probability rating of the system's top-rated hypothesis did not exceed 50%. Overall success was then computed by averaging together the results of the whole series of experiments.

Each experiment consisted of two parts, one in

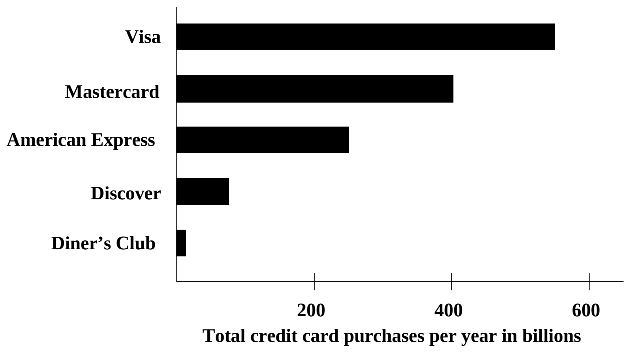


Figure 4: A Graphic from Business Week[3]

which captions were not taken into account in the Bayesian network and one in which the Bayesian network included evidence from captions. Our overall accuracy without the caption evidence was 64.5%, while the inclusion of caption evidence increased accuracy to 79.1% for an absolute increase in accuracy of 14.6% and a relative improvement of 22.6% over the system's accuracy without caption evidence. Thus we conclude that shallow processing of a caption provides evidence that can be effectively utilized in a Bayesian network to recognize the intended message of an information graphic.

Our analysis of the results provides some interesting insights on the role of elements of the caption. There appear to be two primary functions of verbs. The first is to reflect what is in the data, thereby strengthening the message that would be recognized without the caption. One example from our corpus is a graphic with the caption *"Legal immigration to the U.S. has been rising for decades"*. Although the early part of the graphic displays a change from decreasing immigration to a steadily increasing immigration trend, most of the graphic focuses on the decades of increasing immigration and the caption strengthens *increasing trend in immigration* as the intended message of the graphic. If we do not include the caption, our system hypothesizes an *increasing trend* message with a probability of 66.4%; other hypotheses include an intended message that emphasizes the *change in trend* with a probability of 15.3%. However, when the verb *increasing* from the caption is taken into account, the probability of *increasing trend in immigration* being the intended message rises to 97.9%.

The second function of a verb is to focus attention on some aspect of the data. For example, consider the graphic in Figure 4. Without a caption, our system hypothesizes that the graphic is intended to convey the relative rank in billings of different credit card issuers and assigns it a probability of 72.7%. Other possibilities have some probability assigned to them. For example, the intention of conveying that Visa has the highest billings is assigned a probability of 26%. Suppose that the graphic had a caption of *"Billings still lag"*; if the verb *lag* is taken into account, our system hypothesizes an intended message of conveying the credit card issuer whose billings are lowest, namely Diner's Club; the probability assigned to this intention is now 88.4%, and the probability assigned to the intention of conveying the relative rank of different credit card issuers drops to 7.8%. This is because the verb class containing *lag* appeared in our corpus as part of the caption for graphics whose message conveyed an entity with a minimum value, and not with graphics whose message conveyed the relative rank of all the depicted entities. On the other hand, if the caption is *"American Express total billings still lag"* (which is the caption associated with the graphic in our corpus), then we have two pieces of evidence from the caption — the verb *lag*, and the noun *American Express* which matches a label. In this case, the probabilities change dramatically; the hypothesis that the graphic is intended to convey the rank of *American Express* (namely third behind Visa and Mastercard) is assigned a probability of 76% and the probability drops to 24% that the graphic is intended to convey that Diner's Club has the lowest billings. This is not surprising. The presence of the noun *American Express* in the caption makes that entity salient and is very strong evidence that the intended message places an emphasis on *American Express*, thus significantly affecting the probabilities of the different hypotheses. On the other hand, the verb class containing *lag* occurred both in the caption of graphics whose message was judged to convey the entity with the minimum value and in the caption of graphics

[3]This is a slight variation of the graphic from Business Week. In the Business Week graphic, the labels sometimes appear on the bars and sometimes next to them, and the heading for the dependent axis appears in the empty white space of the graphic instead of below the values on the horizontal axis as we show it. Our vision system does not yet have heuristics for recognizing non-standard placement of labels and axis headings.

that conveyed an entity ranked behind some others. Therefore, conveying the entity with minimum value is still assigned a non-negligible probability.

7 Future Work

It is rare that a caption contains more than one verb class; when it does happen, our current system by default uses the first one that appears. We need to examine how to handle the occurrence of multiple verb classes in a caption. Occasionally, labels in the graphic appear differently in the caption. An example is *DJIA* (for Dow Jones Industrial Average) that occurs in one graphic as a label but appears as *Dow* in the caption. We need to investigate resolving such coreferences.

We currently limit ourselves to recognizing what appears to be the primary communicative intention of an information graphic; in the future we will also consider secondary intentions. We will also extend our work to other kinds of information graphics such as line graphs and pie charts, and to complex graphics, such as grouped and composite bar charts.

8 Summary

To our knowledge, our project is the first to investigate the problem of understanding the intended message of an information graphic. This paper has focused on the communicative evidence present in an information graphic and how it can be used in a probabilistic framework to reason about the graphic's intended message. The paper has given particular attention to evidence provided by the graphic's caption. Our corpus study showed that about half of all captions contain some evidence that contributes to understanding the graphic's message, but that fully understanding captions is a difficult problem. We presented a strategy for extracting evidence from a shallow analysis of the caption and utilizing it, along with communicative signals from the graphic itself, in a Bayesian network that hypothesizes the intended message of an information graphic, and our results demonstrate the effectiveness of our methodology. Our research is part of a larger project aimed at providing alternative access to information graphics for individuals with sight impairments.

References

J. Ang, R. Dhillon, A. Krupski, E. Shriberg, and A. Stolcke. 2002. Prosody-based automatic detection of annoyance and frustration in human-computer dialog. In *Proc. of the Int'l Conf. on Spoken Language Processing (ICSLP)*.

D. Chester and S. Elzer. 2005. Getting computers to see information graphics so users do not have to. To appear in *Proc. of the 15th Int'l Symposium on Methodologies for Intelligent Systems*.

H. Clark. 1996. *Using Language*. Cambridge University Press.

M. Corio and G. Lapalme. 1999. Generation of texts for information graphics. In *Proc. of the 7th European Workshop on Natural Language Generation*, 49–58.

S. Elzer, S. Carberry, N. Green, and J. Hoffman. 2004. Incorporating perceptual task effort into the recognition of intention in information graphics. In *Proceedings of the 3rd Int'l Conference on Diagrams*, LNAI 2980, 255–270.

S. Elzer, S. Carberry, I. Zukerman, D. Chester, N. Green, S. Demir. 2005. A probabilistic framework for recognizing intention in information graphics. To appear in *Proceedings of the Int'l Joint Conf. on AI* (IJCAI).

R. Futrelle and N. Nikolakis. 1995. Efficient analysis of complex diagrams using constraint-based parsing. In *Proc. of the Third International Conference on Document Analysis and Recognition*.

R. Futrelle. 1999. Summarization of diagrams in documents. In I. Mani and M. Maybury, editors, *Advances in Automated Text Summarization*. MIT Press.

Nancy Green, Giuseppe Carenini, Stephan Kerpedjiev, Joe Mattis, Johanna Moore, and Steven Roth. Autobrief: an experimental system for the automatic generation of briefings in integrated text and information graphics. *International Journal of Human-Computer Studies*, 61(1):32–70, 2004.

H. P. Grice. 1969. Utterer's Meaning and Intentions. *Philosophical Review*, 68:147–177.

M. Huber, E. Durfee, and M. Wellman. 1994. The automated mapping of plans for plan recognition. In *Proc. of Uncertainty in AI*, 344–351.

S. Kerpedjiev and S. Roth. 2000. Mapping communicative goals into conceptual tasks to generate graphics in discourse. In *Proc. of Int. Conf. on Intelligent User Interfaces*, 60–67.

J. Yu, J. Hunter, E. Reiter, and S. Sripada. 2002. Recognising visual patterns to communicate gas turbine time-series data. In *ES2002*, 105–118.

Scaling up from Dialogue to Multilogue: some principles and benchmarks

Jonathan Ginzburg and Raquel Fernández
Dept of Computer Science
King's College, London
The Strand, London WC2R 2LS
UK
{ginzburg,raquel}@dcs.kcl.ac.uk

Abstract

The paper considers how to scale up dialogue protocols to multilogue, settings with multiple conversationalists. We extract two benchmarks to evaluate scaled up protocols based on the long distance resolution possibilities of non-sentential utterances in dialogue and multilogue in the British National Corpus. In light of these benchmarks, we then consider three possible transformations to dialogue protocols, formulated within an issue-based approach to dialogue management. We show that one such transformation yields protocols for querying and assertion that fulfill these benchmarks.

1 Introduction

The development of dialogue systems in which a human agent interacts using natural language with a computational system is by now a flourishing domain (see e.g. (NLE, 2003)), buttressed by an increasing theoretical and experimental literature on the properties of dialogue (see e.g. recent work in the SEMDIAL and SIGDIAL conferences). In contrast, the development of *multilogue* systems, in which conversation with 3 or more participants ensue—is still in its early stages, as is the theoretical and experimental study of multilogue. *The* fundamental issue in tackling multilogue is: how can mechanisms motivated for dialogue (e.g. information states, protocols, update rules etc) be scaled up to multilogue?

In this paper we extract from a conversational corpus, the British National Corpus (BNC), several benchmarks that characterize dialogue and multilogue interaction. These are based on the resolution possibilities of non-sentential utterances (NSUs). We then use these benchmarks to evaluate certain general transformations whose application to a dialogue interaction system yield a system appropriate for multilogue.

There are of course various plausible views of the relation between dialogue and multilogue. One possible approach to take is to view multilogue as a sequence of dialogues. Something like this approach seems to be adopted in the literature on communication between autonomous software agents. However, even though many situations considered in multiagent systems do involve more than two agents, most interaction protocols are designed only for two participants at a time. This is the case of the protocol specifications provided by FIPA (Foundation for Intelligent Physical Agents) for agent communication language messages (FIPA, 2003). The FIPA interaction protocols (IP) are most typically designed for two participants, an initiator and a responder . Some IPs permit the broadcasting of a message to a group of addressees, and the reception of multiple responses by the original initiator (see most particularly the Contract Net IP). However, even though more than two agents participate in the communicative process, as (Dignum and Vreeswijk, 2003) point out, such conversations can not be considered multilogue, but rather a number of parallel dialogues.

The Mission Rehearsal Exercise (MRE) Project (Traum and Rickel, 2002), one of the largest multilogue systems developed hitherto, is a virtual reality environment where multiple partners (including humans and other autonomous agents) engage in multi-conversation situations. The MRE is underpinned by an approach to the modelling of interaction in terms of obligations that different utterance types bring about originally proposed for dialogue (see e.g. (Matheson et al. , 2000)). In particular, this includes a model of the grounding process (Clark, 1996) that involves recognition and construction of common ground units (CGUs) (see (Traum, 2003)). Modelling of obligations and grounding becomes more complex when considering multilogue situations. The model of grounding implemented in the MRE project can only be used in cases where there is a single initiator and responder. It is not clear what the model should be for multiple addressees: should the contents be considered grounded when any of the addressees has acknowledged them? Should evidence of understanding be required from every addressee?

Since their resolution is almost wholly reliant on context, non sentential utterances provide a large testbed concerning the structure of both dialogue and multilogue. In section 2 we present data from the British National Corpus (BNC) concerning the resolution of NSUs in dialogue and multilogue. The main focus of this data is with the distance between antecedent and fragment. We use this to extract certain benchmarks concerning multilogue interaction. Thus, acknowledgement and acceptance markers (e.g. 'mmh', 'yeah') are resolved with reference to an utterance (assertion) which they ground (accept). The data we provide shows that acknowledgements in multilogue, as in dialogue, are adjacent to their antecedent. This provides evidence that, in general, a single addressee serves to signal grounding. In contrast, BNC data indicates the prevalence in multilogue of short answers that are resolved using material from an antecedent question located several turns back, whereas in dialogue short answers are generally adjacent to their antecedent. This provides evidence against reducing querying interaction in multilogue to a sequence of dialogues. We show that long distance short answers are a stable phenomenon for multilogue involving both small ($\leq$5 persons) and large ($>$ 5 persons) groups, despite the apparently declining interactivity with increasing group size flagged in experimental work (see (Fay et al., 2000)).

In section 3 we sketch the basic principles of issue based dialogue management which we use as a basis for our subsequent investigations of multilogue interaction. This will include information states and formulation of protocols for querying and assertion in dialogue. In section 4 we consider three possible transformations on dialogue protocols into multilogue protocols. These transformations are entirely general in nature and could be applied to protocols stated in whatever specification language. We evaluate the protocols that are generated by these transformations with reference to the benchmarks extracted in section 2. In particular, we show that one such transformation, dubbed Add Side Participants(ASP), yields protocols for querying and assertion that fulfill these benchmarks. Finally, section 5 provides some conclusions and pointers to future work.

2 Long Distance Resolution of NSUs in Dialogue and Multilogue: some benchmarks

The work we present in this paper is based on empirical evidence provided by corpus data extracted from the British National Corpus (BNC).

2.1 The Corpus

Our current corpus is a sub-portion of the BNC conversational transcripts consisting of 14,315 sentences. The corpus was created by randomly excerpting a 200-speaker-turn section from 54 BNC files. Of these files, 29 are transcripts of conversations between two dialogue participants, and 25 files are multilogue transcripts.

A total of 1285 NSUs were found in our sub-corpus. Table 1 shows the raw counts of NSUs found in the dialogue and multilogue transcripts, respectively.

	NSUs	BNC files
Dialogue	709	29
Multilogue	576	25
Total	1285	54

Table 1: Total of NSUs in Dialogue and Multilogue

All NSUs encountered within the corpus were classified according to the NSU typology presented in (Fernández and Ginzburg, 2002). Additionally, the distance from their antecedent was measured.[1] Table 2 shows the distribution of NSU categories and their antecedent separation distance. The classes of NSU which feature in our discussion below are boldfaced.

The BNC annotation includes tagging of units approximating to sentences, as identified by the CLAWS segmentation scheme (Garside, 1987). Each sentence unit is assigned an identifier number. By default it is assumed that sentences are non-overlapping and that their numeration indicates temporal sequence. When this is not the case because speakers overlap, the tagging scheme encodes synchronous speech by means of an alignment map used to synchronize points within the transcription. However, even though information about simultaneous speech is available, overlapping sentences are annotated with different sentence numbers.

In order to be able to measure the distance between the NSUs encountered and their antecedents, all instances were tagged with the sentence number of their antecedent utterance. The distance we report is therefore measured in terms of sentence numbers. It should however be noted that taking into account synchronous speech would not change the data reported in Table 2 in any significant

[1]This classification was done by one expert annotator. To assess its reliability a pilot study of the taxonomy was performed using two additional non-expert coders. These annotated 50 randomly selected NSUs (containing a minimum of 2 instances of each NSU class, as labelled by the expert annotator.). The agreement achieved by the three coders is reasonably good, yielding a kappa score κ = 0.76. We also assessed the accuracy of the coders' choices in choosing the antecedent utterance using the expert annotator's annotation as a gold standard. Given this, one coder's accuracy was 92%, whereas the other coder's was 96%.

NSU Class	Example	Total	Distance						
			1	2	3	4	5	6	>6
Acknowledgment	*Mm mm.*	595	578	15	2				
Short Answer	*Ballet shoes.*	188	104	21	17	5	5	8	28
Affirmative Answer	*Yes.*	109	104	4			1		
Clarification Ellipsis	*John?*	92	76	13	2	1			
Repeated Ack.	*His boss, right.*	86	81	2	3				
Rejection	*No.*	50	49	1					
Factual Modifier	*Brilliant!*	27	23	2	1	1			
Repeated Aff. Ans.	*Very far, yes.*	26	25	1					
Helpful Rejection	*No, my aunt.*	24	18	5		1			
Check Question	*Okay?*	22	15	7					
Filler	*... a cough.*	18	16	1		1			
Bare Mod. Phrase	*On the desk.*	16	11	4			1		
Sluice	*When?*	11	10	1					
Prop. Modifier	*Probably.*	11	10	1					
Conjunction Phrase	*Or a mirror.*	10	5	4	1				
	Total	1285	1125	82	26	9	7	8	28
	Percentage	100	87.6	6.3	2	0.6	0.5	0.6	2.1

Table 2: NSUs sorted by Class and Distance

way, as manual examination of all NSUs at more than distance 3 reveals that the transcription portion between antecedent and NSU does not contain any completely synchronous sentences in such cases.

In the examples throughout the paper we shall use italics to indicate speech overlap. When italics are not used, utterances take place sequentially.

2.2 NSU-Antecedent Separation Distance

The last row in Table 2 shows the distribution of NSU-antecedent separation distances as percentages of the total of NSUs found. This allows us to see that about 87% of NSUs have a distance of 1 sentence (i.e. the antecedent was the immediately preceding sentence), and that the vast majority (about 96%) have a distance of 3 sentences or less.

Although the proportion of NSUs found in dialogue and multilogue is roughly the same (see Table 1 above), when taking into account the distance of NSUs from their antecedent, the proportion of long distance NSUs in multilogue increases radically: the longer the distance, the higher the proportion of NSUs that were found in multilogue. In fact, as Table 3 shows, NSUs that have a distance of 7 sentences or more appear exclusively in multilogue transcripts. These differences are significant (χ^2 = 62.24, $p \leq 0.001$).

Adjacency of grounding and affirmation utterances The data in table 2 highlights a fundamental characteristic of the remaining majoritarian classes of NSUs, Ack(nowledgements), Affirmative Answer, CE (clarification ellipsis), Repeated Ack(nowledgements), and Rejection. These are used either in grounding interaction, or to affirm/reject propositions.[2] The overwhelming adjacency to their antecedent underlines the locality of these interactions.

Long distance potential for short answers One striking result exhibited in Table 2 is the uneven distribution of long distance NSUs across categories. With a few exceptions, NSUs that have a distance of 3 sentences or more are exclusively short answers. Not only is the long distance phenomenon almost exclusively restricted to short answers, but the frequency of long distance short answers stands in strong contrast to the other NSUs classes; indeed, over 44% of short answers have more than distance 1, and over 24% have distance 4 or more, like the last answer in the following example:

(1)		
	Allan:	How much do you think?
	Cynthia:	Three hundred pounds.
	Sue:	More.
	Cynthia:	A thousand pounds.
	Allan:	More.
	Unknown:	<unclear>
	Allan:	Eleven hundred quid apparently. [BNC, G4X]

Long distance short answers primarily a multilogue effect Table 4 shows the total number of short answers found in dialogue and multilogue respectively, and the proportions sorted by distance over those totals:

From this it emerges that short answers are more common in multilogue than in dialogue—134(71%) v.

[2]Acknowledgements and acceptances are, in principle, distinct acts: the former involves indication that an utterance has been understood, whereas the latter that an assertion is accepted. In practice, though, acknowledgements in the form of NSUs commonly simultaneously signal acceptances. Given this, corpus studies of NSUs (e.g. (Fernández and Ginzburg, 2002)) often conflate the two.

Distance	1	2	3	4	5	6	>6
Dialogue	658 (59%)	37 (45%)	11 (45%)	1 (12%)	1 (14%)	1 (13%)	0 (0%)
Multilogue	467 (41%)	45 (55%)	15 (55%)	8 (88%)	6 (86%)	7 (87%)	28 (100%)

Table 3: NSUs in dialogue and multilogue sorted by distance

Short Answers	Total #	1	2	3	> 3
Dialogue	54	82	9	9	0
Multilogue	134	44	11	8	37

Table 4: % over the totals found in dialogue and multilogue

54(29%). Also, the distance pattern exhibited by these two groups is strikingly different: Only 18% of short answers found in dialogue have a distance of more than 1 sentence, with all of them having a distance of at most 3, like the short answer in (2).

(2)

Malcolm:	[...] cos what's three hundred and sixty divided by seven?
Anon 1:	I don't know.
Malcolm:	Yes I don't know either!
Anon 1:	Fifty four point fifty one point four. [BNC, KND]

This dialogue/multilogue asymmetry argues against reductive views of multilogue as sequential dialogue.

Long Distance short answers and group size As Table 4 shows, all short answers at more than distance 3 appear in multilogues. Following (Fay et al., 2000), we distinguish between small groups (those with 3 to 5 participants) and large groups (those with more than 5 participants). The size of the group is determined by the amount of participants that are active when a particular short answer is uttered. We consider active participants those that have made a contribution within a window of 30 turns back from the turn where the short answer was uttered.

Table 5 shows the distribution of long distance short answers (distance > 3) in small and large groups respectively. This indicates that long distance short answers are significantly more frequent in large groups ($\chi^2 = 22.17$, $p \leq 0.001$), though still reasonably common in small groups. A pragmatic account correlating group size and frequency of long distance short answers is offered in the final paragraph of section 3.

Group Size	d > 3	d ≤ 3	Total
≤ 5	20 (21.5%)	73 (78.5%)	93
> 5	26 (63%)	15 (37%)	41

Table 5: Long distance short answers in small and large groups

Large group multilogues in the corpus are all transcripts of tutorials, training sessions or seminars, which exhibit a rather particular structure. The general pattern involves a question being asked by the tutor or session leader, the other participants then taking turns to answer that question. The tutor or leader acts as turn manager. She assigns the turn explicitly usually by addressing the participants by their name without need to repeat the question under discussion. An example is shown in (3):

(3)

Anon1:	How important is those three components and what value would you put on them [...]
Anon3:	Tone forty five. Body language thirty .
Anon1:	Thank you.
Anon4:	Oh.
Anon1:	Melanie.
Anon5:	twenty five.
Anon1:	Yes.
Anon5:	Tone of voice twenty five. [BNC, JYM]

Small group multilogues on the other hand have a more unconstrained structure: after a question is asked, the participants tend to answer freely. Answers by different participants can follow one after the other without explicit acknowledgements nor turn management, like in (4):.

(4)

Anon 1:	How about finance then? <pause>
Unknown 1:	Corruption
Unknown 2:	Risk <pause dur=30>
Unknown 3:	Wage claims <pause dur=18>

2.3 Two Benchmarks of multilogue

The data we have seen above leads in particular to the following two benchmarks protocols for querying, assertion, and grounding interaction in multilogue:

(5) a. **Multilogue Long Distance short answers (MLDSA)**: querying protocols for multilogue must license short answers an unbounded number of turns from the original query.

b. **Multilogue adjacency of grounding/acceptance (MAG)**: assertion and grounding protocols for multilogue should license grounding/clarification/acceptance moves only adjacently to their antecedent utterance.

MLDSA and MAG have a somewhat different status: whereas MLDSA is a direct generalization from the data, MAG is a negative constraint, posited given the paucity of positive instances. As such MAG is more open to doubt and we shall treat it as such in the sequel.

3 Issue based Dialogue Management: basic principles

In this section we outline some of the basic principles of Issue-based Dialogue Management, which we use as a basis for our subsequent investigations of multilogue interaction.

Information States We assume information states of the kind developed in the KoS framework (e.g. (Ginzburg, 1996, forthcoming), (Larsson, 2002)) and implemented in systems such as GODIS, IBIS, and CLARIE (see e.g. (Larsson, 2002; Purver, 2004)). On this view each dialogue participant's view of the common ground, their Dialogue Gameboard (DGB), is structured by a number of attributes including the following three: FACTS: a set of facts representing the shared assumptions of the CPs, LatestMove: the most recent grounded move, and QUD ('questions under discussion'): a partially ordered set—often taken to be structured as a stack—consisting of the currently discussable questions.

Querying and Assertion Both querying and assertion involve a question becoming maximal in the querier/asserter's QUD:[3] the posed question q for a query where q is posed, the polar question $p?$ for an assertion where p is asserted. Roughly, the responder can subsequently either choose to start a discussion (of q or $p?$) or, in the case of assertion, to update her FACTS structure with p. A dialogue participant can downdate $q/p?$ from QUD when, as far as her (not necessarily public) goals dictate, sufficient information has been accumulated in FACTS. The querying/assertion protocols (in their most basic form) are summarized as follows:

(6)

querying	assertion
LatestMove = Ask(A,q)	LatestMove = Assert(A,p)
A: push q onto QUD; release turn;	A: push p? onto QUD; release turn
B: push q onto QUD; take turn; make max-qud–specific; utterance[4] take turn.	B: push p? onto QUD; take turn; Option 1: Discuss p? Option 2: Accept p
	LatestMove = Accept(B,p)
	B: increment FACTS with p; pop p? from QUD;
	A: increment FACTS with p; pop p? from QUD;

Following (Larsson, 2002; Cooper, 2004), one can decompose interaction protocols into *conversational update rules*—functions from DGBs into DGBs using Type Theory with Records (TTR). This allows simple interfacing with the grammar, a Constraint-based Grammar closely modelled on HPSG but formulated in TTR (see (Ginzburg, forthcoming)).

Grounding Interaction Grounding an utterance $u : T$ ('the sign associated with u is of type T') is modelled as involving the following interaction. (a) Addressee B tries to anchor the contextual parameters of T. If successful, B acknowledges u (directly, gesturally or implicitly) and responds to the content of u. (b) If unsuccessful, B poses a Clarification Request (CR), that arises via *utterance coercion* (see (Ginzburg and Cooper, 2001)). For reasons of space we do not formulate an explicit protocol here—the structure of such a protocol resembles the assertion protocol. Our subsequent discussion of assertion can be modified *mutatis mutandis* to grounding.

NSU Resolution We assume the account of NSU resolution developed in (Ginzburg and Sag, 2000). The essential idea they develop is that NSUs get their main predicates from context, specifically via unification with the question that is currently *under discussion*, an entity dubbed the *maximal question under discussion* (MAX-QUD). NSU resolution is, consequently, tied to conversational topic, viz. the MAX-QUD.[5]

Distance effects in dialogue short answers If one assumes QUD to be a stack, this affords the potential for non adjacent short answers in dialogue. These, as discussed in section 2, are relatively infrequent. Two commonly observed *dialogue* conditions will jointly enforce adjacency between short answers and their interrogative antecedents: (a) Questions have a simple, one phrase answer. (b) Questions can be answered immediately, without preparatory or subsequent discussion. For multilogue (or at least certain genres thereof), both these conditions are less likely to be maintained: different CPs can supply different answers, even assuming that relative to each CP there is a simple, one phrase answer. The more CPs there are in a conversation, the smaller their common ground and the more likely the need for clarificatory interaction. A pragmatic account of this type of the frequency of adjacency in dialogue short answers seems clearly preferable to any actual mechanism that would *rule out* long distance short answers. These can be perfectly felicitous—see e.g. example (1) above which

[3]In other words, *pushed onto the stack*, if one assumes QUD is a stack.

[4]An utterance whose content is either a proposition p About max-qud or a question q_1 on which max-qud Depends. For the latter see footnote 7. If one assumes QUD to be a stack, then 'max-qud–specific' will in this case reduce to 'q–specific'. But the more general formulation will be important below.

[5]The resolution of NSUs, on the approach of (Ginzburg and Sag, 2000), involves one other parameter, an antecedent sub-utterance they dub the *salient-utterance* (SAL-UTT). This plays a role similar to the role played by the *parallel element* in higher order unification–based approaches to ellipsis resolution (see e.g. (Pulman, 1997). For current purposes, we limit attention to the MAX-QUD as the nucleus of NSU resolution.

would work fine if the turn uttered by Sue had been uttered by Allan instead. Moreover such a pragmatic account leads to the expectation that the frequency of long distance antecedents is correlated with group size, as indeed indicated by the data in table 5.

4 Scaling up Protocols

(Goffman, 1981) introduced the distinction between *ratified participants* and *overhearers* in a conversation. Within the former are located the speaker and participants whom she takes into account in her utterance design—the intended addressee(s) of a given utterance, as well as *side participants*. In this section we consider three possible principles of protocol extension, each of which can be viewed as adding roles for participants from one of Goffman's categories. We evaluate the protocol that results from the application of each such principle relative to the benchmarks we introduced in section 2.3. Seen in this light, the final principle we consider, Add Side Participants (ASP), arguably, yields the best results. Nonetheless, these three principles would appear to be complementary—the most general protocol for multilogue will involve, minimally, application of all three.[6] We state the principles informally and framework independently as transformations on operational construals of the protocols. In a more extended presentation we will formulate these as functions on TTR conversational update rules.

The simplest principle is Add Overhearers (AOV). This involves adding participants who merely observe the interaction. They keep track of facts concerning a particular interaction, but their context is not facilitated for them to participate:

(7) Given a dialogue protocol π, add roles C$_1$,…,C$_n$ where each C$_i$ is a silent participant: given an utterance u_0 classified as being of type T_0, C$_i$ updates C$_i$.DGB.FACTS with the proposition $u_0 : T_0$.

Applying AOV yields essentially multilogues which are sequences of dialogues. A special case of this are moderated multilogues, where all dialogues involve a designated individual (who is also responsible for turn assignment.). Restricting scaling up to applications of AOV is not sufficient since *inter alia* this will not fulfill the MLDSA benchmark.

A far stronger principle is Duplicate Responders (DR):

(8) Given a dialogue protocol π, add roles C$_1$,…,C$_n$ which duplicate the responder role.

[6]We thank an anonymous reviewer for ACL for convincing us of this point.

Applying DR to the querying protocol yields the following protocol:

(9) Querying with multiple responders

1. LatestMove = Ask(A,q)
2. A: push q onto QUD; release turn
3. Resp$_1$: push q onto QUD; take turn; make max-qud–specific utterance; release turn
4. Resp$_2$: push q onto QUD; take turn; make max-qud–specific utterance; release turn
5. …
6. Resp$_n$: push q onto QUD; take turn; make max-qud–specific utterance; release turn

This yields interactions such as (4) above. The querying protocol in (9) licenses long distance short answers, so satisfies the MLDSA benchmark. On the other hand, the contextual updates it enforces will not enable it to deal with the following (constructed) variant on (4), in other words does not afford responders to comment on previous responders, as opposed to the original querier:

(10) A: Who should we invite for the conference?
B: Svetlanov.
C: No (=Not Svetlanov), Zhdanov
D: No (= Not Zhdanov, $\neq$ Not Svetlanov), Gergev

Applying DR to the assertion protocol will yield the following protocol:

(11) Assertion with multiple responders

1. LatestMove = Assert(A,p)
2. A: push p? onto QUD; release turn
3. Resp$_1$: push p? onto QUD; take turn; $\langle$ Option 1: Discuss p?, Option 2: Accept p $\rangle$
4. Resp$_2$: push p? onto QUD; take turn; $\langle$ Option 1: Discuss p?, Option 2: Accept p $\rangle$
5. …
6. Resp$_n$: push p? onto QUD; take turn; $\langle$ Option 1: Discuss p?, Option 2: Accept p $\rangle$

One arguable problem with this protocol—equally applicable to the corresponding DRed grounding protocol—is that it licences long distance acceptance and is, thus, inconsistent with the MAG benchmark. On the other hand, it is potentially useful for interactions where there is explicitly more than one direct addressee.

A principle intermediate between AOV and DR is Add Side Participants (ASP):

(12) Given a dialogue protocol π, add roles C$_1$,…,C$_n$, which effect the same contextual update as the interaction initiator.

Applying ASP to the dialogue assertion protocol yields the following protocol:

(13) Assertion for a conversation involving $\{$A,B,C$_1$,…,C$_n\}$

1. LatestMove = Assert(A,p)
2. A: push p? onto QUD; release turn
3. C_i: push p? onto QUD;
4. B: push p? onto QUD; take turn; ⟨Option 1: Accept p, Option 2: Discuss p?⟩

(14)
1. LatestMove = Accept(B,p)
2. B: increment FACTS with p; pop p? from QUD;
3. C_i:increment FACTS with p; pop p? from QUD;
4. A: increment FACTS with p; pop p? from QUD;

This protocol satisfies the MAG benchmark in that acceptance is strictly local. This is because it enforces *communal acceptance*—acceptance by one CP can count as acceptance by all other addressees of an assertion. There is an obvious rational motivation for this, given the difficulty of a CP constantly monitoring an entire audience (when this consists of more than one addressee) for acceptance signals—it is well known that the effect of visual access on turn taking is highly significant (Dabbs and Ruback, 1987). It also enforces quick reaction to an assertion—anyone wishing to dissent from p must get their reaction in early i.e. immediately following the assertion since further discussion of $p?$ is not countenanced if acceptance takes place. The latter can happen of course as a consequence of a dissenter not being quick on their feet; on this protocol to accommodate such cases would require some type of backtracking.

Applying ASP to the dialogue querying protocol yields the following protocol:

(15) Querying for a conversation involving $\{$ A,B,C_1,. . . ,C_n $\}$
1. LatestMove = Ask(A,q)
2. A: push q onto QUD; release turn
3. C_i: push q onto QUD;
4. B: push q onto QUD; take turn; make max-qud–specific utterance.

This improves on the DR generated protocol because it does allow responders to comment on previous responders—the context is modified as in the dialogue protocol. Nonetheless, as it stands, this protocol won't fully deal with examples such as (4)—the issue introduced by each successive participant takes precedence given that QUD is assumed to be a stack. This can be remedied by slightly modifying this latter assumption: we will assume that when a question q is pushed onto QUD it doesn't subsume *all* existing questions in QUD, but rather only those on which q does not depend:[7]

(16) q is $\text{QUD}^{mod(dependence)}$ maximal iff for any q_0 in QUD such that $\neg\text{Depend}(q, q_1)$: $q \succ q_0$.

[7] The notion of dependence we assume here is one common in work on questions, e.g. (Ginzburg and Sag, 2000), intuitively corresponding to the notion of 'is a subquestion of'. q_1 depends on q_2 iff any proposition p such that p resolves q_2 also satisfies p is about q_1.

This is conceptually attractive because it reinforces that the order in QUD has an intuitive semantic basis. One effect this has is to ensure that any polar question $p?$ introduced into QUD, whether by an assertion or by a query, subsequent to a wh-question q on which $p?$ depends does not subsume q. Hence, q will remain accessible as an antecedent for NSUs, as long as no new unrelated topic has been introduced. Assuming this modification to QUD is implemented in the above ASP–generated protocols, both MLDSA and MAG benchmarks are fulfilled.

5 Conclusions and Further Work

In this paper we consider how to scale up dialogue protocols to multilogue, settings with multiple conversationalists. We have extracted two benchmarks, MLDSA and MAG, to evaluate scaled up protocols based on the long distance resolution possibilities of NSUs in dialogue and multilogue in the BNC. MLDSA, the requirement that multilogue protocols license long distance short answers, derives from the statistically significant increase in frequency of long distance short answers in multilogue as opposed to dialogue. MAG, the requirement that multilogue protocols enforce adjacency of acceptance and grounding interaction, derives from the overwhelming locality of acceptance/grounding interaction in multilogue, as in dialogue. In light of these benchmarks, we then consider three possible transformations to dialogue protocols formulated within an issue-based approach to dialogue management. Each transformation can be intuited as adding roles that correspond to distinct categories of an audience originally suggested by Goffman. The three transformations would appear to be complementary—it seems reasonable to assume that application of all three (in some formulation) will be needed for wide coverage of multilogue. MLDSA and MAG can be fulfilled within an approach that combines the Add Side Participants transformation on protocols with an independently motivated modification of the structure of QUD from a canonical stack to a stack where maximality is conditioned by issue dependence.

With respect to long distance short answers our account licences their occurrence in dialogue, as in multilogue. We offer a pragmatic account for their low frequency in dialogue, which indeed generalizes to explain a statistically significant correlation we observe between their increased incidence and increasing active participant size. We plan to carry out more detailed work, both corpus–based and experimental, in order to evaluate the status of MAG and, correspondingly to assess just how local acceptance and grounding interaction really are. We also intend to implement multilogue protocols in CLARIE so it can simulate multilogue. We will then evaluate its ability to process NSUs from the BNC.

Acknowledgements

We would like to thank three anonymous ACL reviewers for extremely useful comments, which in particular forced us to rethink some key issues. We would also like to thank Pat Healey, Shalom Lappin, Richard Power, and Matt Purver for discussion, and Zoran Macura and Yo Sato for help in assessing the NSU taxonomy. Earlier versions of this work were presented at colloquia at ITRI, Brighton, and at the Université Paris, 7. The research described here is funded by grant number RES-000-23-0065 from the Economic and Social Research Council of the United Kingdom.

References

Special issue on best practice in spoken language dialogue systems engineering. 2003. *Natural Language Engineering*.

Herbert Clark. 1996. *Using Language*. Cambridge University Press, Cambridge.

Robin Cooper. 2004. A type theoretic approach to information state update in issue based dialogue management. Invited paper, *Catalog'04, the 8th Workshop on the Semantics and Pragmatics of Dialogue*, Pompeu Fabra University, Barcelona.

James Dabbs and R. Barry Ruback. 1987 Dimensions of group process: amount and structure of vocal interaction. *Advances in Experimental Social Psychology* 20, pages 123–169.

Frank P.M. Dignum and Gerard A.W. Vreeswijk. 2003. Towards a testbed for multi-party dialogues. In *Proceedings of the first International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2003)*.

Nicholas Fay, Simon Garrod, and Jean Carletta. 2000. Group discussion as interactive dialogue or serial monologue. *Psychological Science*, pages 481–486.

Raquel Fernández and Jonathan Ginzburg. 2002. Non-sentential utterances: A corpus study. *Traitement automatique des languages. Dialogue*, 43(2):13–42.

FIPA. 2003. The foundation for intelligent physical agents. interaction protocol specifications. http://www.fipa.org.

Roger Garside. 1987. The CLAWS word-tagging system, In Roger Garside et al. editors, *The computational analysis of English: a corpus-based approach*, Longman, Harlow, pages 30–41.

Jonathan Ginzburg and Robin Cooper. 2001. Resolving ellipsis in clarification. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*, Toulouse.

Jonathan Ginzburg and Ivan A. Sag. 2000. *Interrogative Investigations: the form, meaning and use of English Interrogatives*. Number 123 in CSLI Lecture Notes. CSLI Publications, Stanford: California.

Jonathan Ginzburg. (forthcoming). *Semantics and Interaction in Dialogue* CSLI Publications and University of Chicago Press.

Jonathan Ginzburg. 1996. Interrogatives: Questions, facts, and dialogue. In Shalom Lappin, editor, *Handbook of Contemporary Semantic Theory*. Blackwell, Oxford.

Erving Goffman 1981 *Forms of Talk*. University of Pennsylvania Press, Philadelphia.

Staffan Larsson. 2002. *Issue based Dialogue Management*. Ph.D. thesis, Gothenburg University.

Colin Matheson and Massimo Poesio and David Traum. 2000. Modelling Grounding and Discourse Obligations Using Update Rules. *Proceedings of NAACL 2000*, Seattle.

Stephen Pulman. 1997. Focus and higher order unification. *Linguistics and Philosophy*, 20.

Matthew Purver. 2004. *The Theory and Use of Clarification in Dialogue*. Ph.D. thesis, King's College, London.

David Traum and Jeff Rickel. 2002. Embodied agents for multi-party dialogue in immersive virtual world. In *Proceedings of the first International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2002)*, pages 766–773.

David Traum. 2003. Semantics and pragmatics of questions and answers for dialogue agents. In H. Bunt, editor, *Proceedings of the 5th International Workshop on Computational Semantics*, pages 380–394, Tilburg. ITK, Tilburg University.

Implications for Generating Clarification Requests in Task-oriented Dialogues

Verena Rieser
Department of Computational Linguistics
Saarland University
Saarbrücken, D-66041
vrieser@coli.uni-sb.de

Johanna D. Moore
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW, GB
J.Moore@ed.ac.uk

Abstract

Clarification requests (CRs) in conversation ensure and maintain mutual understanding and thus play a crucial role in robust dialogue interaction. In this paper, we describe a corpus study of CRs in task-oriented dialogue and compare our findings to those reported in two prior studies. We find that CR behavior in task-oriented dialogue differs significantly from that in everyday conversation in a number of ways. Moreover, the dialogue type, the modality and the channel quality all influence the decision of when to clarify and at which level of the grounding process. Finally we identify form-function correlations which can inform the generation of CRs.

1 Introduction

Clarification requests in conversation ensure and maintain mutual understanding and thus play a significant role in robust and efficient dialogue interaction. From a theoretical perspective, the *model of grounding* explains how mutual understanding is established. According to Clark (1996), speakers and listeners ground mutual understanding on four levels of coordination in an action ladder, as shown in Table 1.

Level	Speaker S	Listener L
Convers.	S is proposing activity α	L is considering proposal α
Intention	S is signalling that p	L is recognizing that p
Signal	S is presenting signal σ	L is identifying signal σ
Channel	S is executing behavior β	L is attending to behavior β

Table 1: Four levels of grounding

Several current research dialogue systems can detect errors on different levels of grounding (Paek and Horvitz, 2000; Larsson, 2002; Purver, 2004; Schlangen, 2004). However, only the work of Purver (2004) addresses the question of how the source of the error affects the form the CR takes.

In this paper, we investigate the use of form-function mappings derived from human-human dialogues to inform the generation of CRs. We identify the factors that determine which function a CR should take and identify function-form correlations that can be used to guide the automatic generation of CRs.

In Section 2, we discuss the classification schemes used in two recent corpus studies of CRs in human-human dialogue, and assess their applicability to the problem of generating CRs. Section 3 describes the results we obtained by applying the classification scheme of Rodriguez and Schlangen (2004) to the Communicator Corpus (Bennett and Rudnicky, 2002). Section 4 draws general conclusions for generating CRs by comparing our results to those of (Purver et al., 2003) and (Rodriguez and Schlangen, 2004). Section 5 describes the correlations between function and form features that are present in the corpus and their implications for generating CRs.

Attr.	Value	Category	Example
form	`non`	Non-Reprise	*"What did you say?"*
	`wot`	Conventional	*"Sorry?"*
	`frg`	Reprise Fragment	*"Edinburgh?"*
	`lit`	Literal Reprise	*"You want a flight to Edinburgh?"*
	`slu`	Reprise Sluice	*"Where?"*
	`sub`	Wh-substituted Reprise	*"You want a flight where?"*
	`gap`	Gap	*"You want a flight to...?"*
	`fil`	Gap Filler	*"...Edinburgh?"*
	`other`	Other	x
readings	`cla`	Clausal	*"Are you asking/asserting that X?"*
	`con`	Constituent	*"What do you mean by X?"*
	`lex`	Lexical	*"Did you utter X?"*
	`corr`	Correction	*"Did you intend to utter X instead?"*
	`other`	Other	x

Table 2: CR classification scheme by PGH

2 CR Classification Schemes

We now discuss two recently proposed classification schemes for CRs, and assess their usefulness for generating CRs in a spoken dialogue system (SDS).

2.1 Purver, Ginzburg and Healey (PGH)

Purver, Ginzburg and Healey (2003) investigated CRs in the British National Corpus (BNC) (Burnard, 2000). In their annotation scheme, a CR can take seven distinct surface forms and four readings, as shown in Table 2. The examples for the form feature are possible CRs following the statement *"I want a flight to Edinburgh"*. The focus of this classification scheme is to map semantic readings to syntactic surface forms. The *form* feature is defined by its relation to the problematic utterance, i.e., whether a CR reprises the antecedent utterance and to what extent. CRs may take the three different readings as defined by Ginzburg and Cooper (2001), as well as a fourth reading which indicates a correction.

Although PGH report good coverage of the scheme on their subcorpus of the BNC (99%), we found their classification scheme to to be too coarse-grained to prescribe the form that a CR should take. As shown in example 1, Reprise Fragments (RFs), which make up one third of the BNC, are ambiguous in their readings and may also take several surface forms.

(1) I would like to book a flight on Monday.

(a) Monday?
`frg, con/cla`

(b) Which Monday?
`frg, con`

(c) Monday the first?
`frg, con`

(d) The first of May?
`frg, con`

(e) Monday the first or Monday the eighth?
`frg, (exclusive) con`

RFs endorse literal repetitions of part of the problematic utterance (1.a); repetitions with an additional question word (1.b); repetition with further specification (1.c); reformulations (1.d); and alternative questions (1.e)[1].

In addition to being too general to describe such differences, the classification scheme also fails to describe similarities. As noted by (Rodriguez and Schlangen, 2004), PGH provide no feature to describe the extent to which an RF repeats the problematic utterance.

Finally, some phenomena cannot be described at all by the four readings. For example, the readings do not account for non-understanding on the pragmatic level. Furthermore the readings may have several problem sources: the clausal reading may be appropriate where the CR initiator failed to recognise the word acoustically as well as when he failed to resolve the reference. Since we are interested in generating CRs that indicate the source of the error, we need a classification scheme that represents such information.

2.2 Rodriguez and Schlangen (R&S)

Rodriguez and Schlangen (2004) devised a multi-dimensional classification scheme where *form* and

[1]Alternative questions would be interpreted as asking a polar question with an exclusive reading.

function are meta-features taking sub-features as attributes. The *function* feature breaks down into the sub-features *source, severity, extent, reply* and *satisfaction*. The *sources* that might have caused the problem map to the levels as defined by Clark (1996). These sources can also be of different *severity*. The severity can be interpreted as describing the set of possible referents: asking for repetition indicates that no interpretation is available (`cont-rep`); asking for confirmation means that the CR initiator has some kind of hypothesis (`cont-conf`). The *extent* of a problem describes whether the CR points out a problematic element in the problem utterance. The *reply* represents the answer the addressee gives to the CR. The *satisfaction* of the CR-initiator is indicated by whether he renews the request for clarification or not.

The meta-feature *form* describes how the CR is lingustically realised. It describes the *sentence's mood*, whether it is grammatically *complete*, the *relation to the antecedent*, and the *boundary tone*. According to R&S's classification scheme our illustrative example would be annotated as follows[2]:

(2) I would like to book a flight on Monday.

(a) Monday?

```
mood: decl
completeness: partial
rel-antecedent: repet
source: acous/np-ref
severity: cont-repet
extent: yes
```

(b) Which Monday?

```
mood: wh-question
completeness: partial
rel-antecedent: addition
source: np-ref
severity: cont-repet
extent: yes
```

(c) Monday the first?

```
mood: decl
completeness: partial
rel-antecedent: addition
source: np-ref
severity: cont-conf
extent: yes
```

(d) The first of May?

```
mood: decl
completeness: partial
rel-antecedent: reformul
source: np-ref
severity: cont-conf
extent: yes
```

(d) Monday the first or Monday the eighth?

```
mood: alt-q
completeness: partial
rel-antecedent: addition
source: np-ref
severity: cont-repet
extent: yes
```

[2]The source features answer and satisfaction are ignored as they depend on how the dialogue continues. The interpretation of the source is dependent on the reply to the CR. Therefore all possible interpretations are listed.

In R&S's classification scheme, ambiguities about CRs having different sources cannot be resolved entirely as example (2.a) shows. However, in contrast to PGH, the overall approach is a different one: instead of explaining causes of CRs within a theoretic-semantic model (as the three different readings of Ginzburg and Cooper (2001) do), they infer the interpretation of the CR from the context. Ambiguities get resolved by the reply of the addressee and the satisfaction of the CR initiator indicates the "mutually agreed interpretation" .

R&S's multi-dimensional CR description allows the fine-grained distinctions needed to generate natural CRs to be made. For example, PGH's general category of RFs can be made more specific via the values for the feature *relation to antecedent*. In addition, the *form* feature is not restricted to syntax; it includes features such as intonation and coherence, which are useful for generating the surface form of CRs. Furthermore, the multi-dimensional *function* feature allows us to describe information relevant to generating CRs that is typically available in dialogue systems, such as the level of confidence in the hypothesis and the problem source.

3 CRs in the Communicator Corpus

3.1 Material and Method

Material: We annotated the human-human travel reservation dialogues available as part of the Carnegie Mellon Communicator Corpus (Bennett and Rudnicky, 2002) because we were interested in studying naturally occurring CRs in task-oriented dialogue. In these dialogues, an experienced travel agent is making reservations for trips that people in the Carnegie Mellon Speech Group were taking in the upcoming months. The corpus comprises 31 dialogues of transcribed telephone speech, with 2098 dialogue turns and 19395 words.

$$\left[\begin{array}{ll}
\textit{form:} & \left[\begin{array}{l}
\texttt{distance-src: } \left\{ \texttt{1} \mid \texttt{2} \mid \texttt{3} \mid \texttt{4} \mid \texttt{5} \mid \texttt{more} \right\} \\
\texttt{mood: } \left\{ \texttt{none} \mid \texttt{decl} \mid \texttt{polar-q} \mid \texttt{wh-q} \mid \texttt{alt-q} \mid \texttt{imp} \mid \texttt{other} \right\} \\
\texttt{form: } \left\{ \texttt{none} \mid \texttt{particle} \mid \texttt{partial} \mid \texttt{complete} \right\} \\
\texttt{relation-antecedent: } \left\{ \texttt{none} \mid \texttt{add} \mid \texttt{repet} \mid \texttt{repet-add} \mid \texttt{reformul} \mid \texttt{indep} \right\} \\
\texttt{boundary-tone: } \left\{ \texttt{none} \mid \texttt{rising} \mid \texttt{falling} \mid \texttt{no-appl} \right\}
\end{array}\right] \\
\textit{function:} & \left[\begin{array}{l}
\texttt{source: } \left\{ \begin{array}{l} \texttt{none} \mid \texttt{acous} \mid \texttt{lex} \mid \texttt{parsing} \mid \texttt{np-ref} \mid \texttt{deitic-ref} \mid \texttt{act-ref} \mid \\ \texttt{int+eval} \mid \texttt{relevance} \mid \texttt{belief} \mid \texttt{ambiguity} \mid \texttt{scr-several} \end{array} \right\} \\
\texttt{extent: } \left\{ \texttt{none} \mid \texttt{fragment} \mid \texttt{whole} \right\} \\
\texttt{severity: } \left\{ \texttt{none} \mid \texttt{cont-conf} \mid \texttt{cont-rep} \mid \texttt{cont-disamb} \mid \texttt{no-react} \right\} \\
\texttt{answer: } \left\{ \begin{array}{l} \texttt{none} \mid \texttt{ans-repet} \mid \texttt{ans-y/n} \mid \texttt{ans-reformul} \mid \texttt{ans-elab} \mid \\ \texttt{ans-w-defin} \mid \texttt{no-react} \end{array} \right\} \\
\texttt{satisfaction: } \left\{ \texttt{none} \mid \texttt{happy-yes} \mid \texttt{happy-no} \mid \texttt{happy-ambig} \right\}
\end{array}\right]
\end{array}\right]$$

Figure 1: CR classification scheme

Annotation Scheme: Our annotation scheme, shown in Figure 1, is an extention of the R&S scheme described in the previous section. R&S's scheme was devised for and tested on the Bielefeld Corpus of German task-oriented dialogues about joint problem solving.[3] To annotate the Communicator Corpus we extended the scheme in the following ways. First, we found the need to distinguish CRs that consist only of newly added information, as in example 3, from those that add information while also repeating part of the utterance to be clarified, as in 4. We augmented the scheme to allow two distinct values for the *form* feature `relation-antecedent`, `add` for cases like 3 and `repet-add` for cases like 4.

(3) **Cust:** What is the last flight I could come back on?
Agent: On the 29th of March?

(4) **Cust:** I'll be returning on Thursday the fifth.
Agent: The fifth of February?

To the function feature *source* we added the values `belief` to cover CRs like 5 and `ambiguity refinement` to cover CRs like 6.

(5) **Agent:** You need a visa.
Cust: I *do* need one?
Agent: Yes you do.

(6) **Agent:** Okay I have two options ... with Hertz ... if not they do have a lower rate with Budget and that is fifty one dollars.
Cust: Per day?
Agent: Per day um mm.

Finally, following Gabsdil (2003) we introduced an additional value for *severity*, `cont-disamb`, to cover CRs that request disambiguation when more than one interpretation is available.

Method: We first identified turns containing CRs, and then annotated them with *form* and *function* features. It is not always possible to identify CRs from the utterance alone. Frequently, context (e.g., the reaction of the addressee) or intonation is required to distinguish a CR from other feedback strategies, such as positive feedback. See (Rieser, 2004) for a detailed discussion. The annotation was only performed once. The coding scheme is a slight variation of R&S, which has been shown relaiable with Kappa of 0.7 for identifying source.

3.2 Forms and Functions of CRs in the Communicator Corpus

The human-human dialogues in the Communicator Corpus contain 98 CRs in 2098 dialogue turns (4.6%).

Forms: The frequencies for the values of the individual *form* features are shown in Table 3. The most frequent type of CRs were *partial declarative questions*, which combine the mood value `declarative` and the completeness value `partial`.[4] These account for 53.1% of the CRs in the corpus. Moreover, four of the five most frequent surface forms of CRs in the Communicator Corpus differ only in the value for the feature `relation-antecedent`. They are partial declaratives with rising boundary tone, that either reformulate (7.1%) the problematic utterance, repeat

[3] `http://sfb360.uni-bielefeld.de`

[4] Declarative questions cover "all cases of non-interrogative word-order, i.e., both declarative sentences and fragments" (Rodriguez and Schlangen, 2004).

Feature	Value	Freq. (%)
Mood	declarative	65
	polar	21
	wh-question	7
	other	7
Completeness	partial	58
	complete	38
	other	4
Relation antecedent	rep-add	27
	independent	21
	reformulation	19
	repetition	18
	addition	10
	other	5
Boundary tone	rising	74
	falling	22
	other	4

Table 3: Distribution of values for the *form* features

Feature	Value	Freq. (%)
Source	np-reference	40
	acoustic	31
	intention	8
	belief	6
	ambiguity	4
	contact	4
	others	3
	relevance	2
	several	2
Extent	yes	80
	no	20
Severity	confirmation	73
	repetition	20
	other	7
Answer	y/n answer	64
	other	15
	elaboration	13
	no reaction	6

Table 4: Distribution of values for the *function* features

the problematic constituent (11.2%), add only new information (7.1%), or repeat the problematic constituent and add new information (10.2%). The fifth most frequent type is conventional CRs (10.2%).[5]

Functions: The distributions of the *function* features are given in Figure 4. The most frequent source of problems was np-reference. Next most frequent were acoustic problems, possibly due to the poor channel quality. Third were CRs that enquire about intention. As indicated by the feature *extent*, almost 80% of CRs point out a specific element of the problematic utterance. The features *severity* and *answer* illustrate that most of the time CRs request confirmation of an hypothesis (73.5%) with a yes-no-answer (64.3%). The majority of the provided answers were satisfying, which means that the addressee tends to interpret the CR correctly and answers collaboratively. Only 6.1% of CRs failed to elicit a response.

4 CRs in Task-oriented Dialogue

4.1 Comparison

In order to determine whether there are differences as regards CRs between task-oriented dialogues and everyday conversations, we compared our results to those of PGH's study on the BNC and those of R&S on the Bielefeld Corpus. The BNC contains a 10 million word sub-corpus of English dialogue transcriptions about topics of general interest. PGH analysed a portion consisting of ca. 10,600 turns, ca. 150,000 words. R&S annotated 22 dialogues from the Bielefeld Corpus, consisting of ca. 3962 turns, ca. 36,000 words.

The major differences in the feature distributions are listed in Table 5. We found that there are no significant differences between the feature distributions for the Communicator and Bielefeld corpora, but that the differences between Communicator and BNC, and Bielefeld and BNC are significant at the levels indicated in Table 5 using Pearson's χ^2. The differences between dialogues of different types suggest that there is a different grounding strategy. In task-oriented dialogues we see a trade-off between avoiding misunderstanding and keeping the conversation as efficient as possible. The hypothesis that grounding in task-oriented dialogues is more *cautious* is supported by the following facts (as shown by the figures in Table 5):

- CRs are more frequent in task-oriented dialogues.
- The overwhelming majority of CRs directly follow the problematic utterance.

[5] Conventional forms are *"Excuse me?"*, *"Pardon?"*, etc.

	Corpus		
Feature	Communicator	Bielefeld	BNC
CRs	98	230	418
frequency	4.6%	5.8%***	3.9%
distance-src=1	92.8%*	94.8%***	84.4%
no-react	6.1%*	8.7%**	17.0%
cont-conf	73.5%***	61.7%***	46.6%
partial	58.2%**	76.5%***	42.4%
independent	21.4%***	9.6%***	44.2%
cont-rep	19.8%***	14.8%***	39.5%
y/n-answer	64.3%	44.8%	n/a

Table 5: Comparison of CR *forms* in everyday vs. task-oriented corpora (* denotes $p < .05$, ** is $p < .01$, *** is $p < .005$.)

	Corpus		
Feature	Communicator	Bielefeld	Significance
contact	4.1%	0 inst	n/a
acoustic	30.6%	11.7%	***
lexical	1 inst	1 inst	n/a
parsing	1 inst	0 inst	n/a
np-ref	39.8%	24.4%	**
deict-ref	1 inst	27.4%	***
ambiguity	4.1%	not eval.	n/a
belief	6.1%	not eval.	n/a
relevance	2.1%	not eval.	n/a
intention	8.2%	22.2%	**
several	2.0%	14.3%	***

Table 6: Comparison of CR problem sources in task-oriented corpora

- CRs in everyday conversation fail to elicit a response nearly three times as often.[6]
- Even though dialogue participants seem to have strong hypotheses, they frequently confirm them.

Although grounding is more cautious in task-oriented dialogues, the dialogue participants try to keep the dialogue as *efficient* as possible:

- Most CRs are partial in form.
- Most of the CRs point out one specific element (with only a minority being independent as shown in Table 5). Therefore, in task-oriented dialogues, CRs locate the understanding problem directly and give partial credit for what was understood.
- In task-oriented dialogues, the CR-initiator asks to confirm an hypothesis about what he understood rather than asking the other dialogue participant to repeat her utterance.
- The addressee prefers to give a short y/n answer in most cases.

Comparing error sources in the two task-oriented corpora, we found a number of differences as shown in Table 6. In particular:

[6]Another factor that might account for these differences is that the BNC contains multi-party conversations, and questions in multi-party conversations may be less likely to receive responses. Furthermore, due to the poor recording quality of the BNC, many utterances are marked as “not interpretable”, which could also lower the response rate.

- *Dialogue type*: Belief and ambiguity refinement do not seem to be a source of problems in joint problem solving dialogues, as R&S did not include them in their annotation scheme. For CRs in information seeking these features need to be added to explain quite frequent phenomena. As shown in Table 6, 10.2% of CRs were in one of these two classes.
- *Modality*: Deictic reference resolution causes many more understanding difficulties in dialogues where people have a shared point of view than in telephone communication (Bielefeld: most frequent problem source; Communicator: one instance detected). Furthermore, in the Bielefeld Corpus, people tend to formulate more fragmentary sentences. In environments where people have a shared point of view, complete sentences can be avoided by using non-verbal communication channels. Finally, we see that establishing contact is more of a problem when speech is the only modality available.
- *Channel quality*: Acoustic problems are much more likely in the Communicator Corpus.

These results indicate that the decision process for grounding needs to consider the modality, the domain, and the communication channel. Similar extensions to the grounding model are suggested by (Traum, 1999).

4.2 Consequences for Generation

The similarities and differences detected can be used to give recommendations for generating CRs. In terms of when to initiate a CR, we can state that clarification should not be postponed, and immediate, local management of uncertainty is critical. This view is also supported by observations of how non-native speakers handle non-understanding (Paek, 2003).

Furthermore, for task-oriented dialogues the system should present an hypothesis to be confirmed, rather than ask for repetition. Our data suggests that, when they are confronted with uncertainty, humans tend to build up hypotheses from the dialogue history and from their world knowledge. For example, when the customer specified a date without a month, the travel agent would propose the most reasonable hypothesis instead of asking a wh-question. It is interesting to note that Skantze (2003) found that users are more satisfied if the system "hides" its recognition problem by asking a task-related question to help to confirm the hypothesis, rather than explicitly indicating non-understanding.

5 Correlations between Function and Form: How to say it?

Once the dialogue system has decided on the *function* features, it must find a corresponding surface form to be generated. Many forms are indeed related to the function as shown in Table 7, where we present a significance analysis using Pearson's χ^2 (with Yates correction).

Source: We found that the relation to the antecedent seems to distinguish fairly reliably between CRs clarifying reference and those clarifying acoustic understanding. In the Communicator Corpus, for acoustic problems the CR-initiator tends to repeat the problematic part literally, while reference problems trigger a reformulation or a repetition with addition. For both problem sources, partial declarative questions are preferred. These findings are also supported by R&S. For the first level of non-understanding, the inability to establish contact, complete polar questions with no relation to the antecedent are formulated, e.g., ”*Are you there?*”.

Severity: The severity indicates how much was understood, i.e., whether the CR initiator asks to confirm an hypothesis or to repeat the antecedent utterance. The severity of an error strongly correlates with the sentence mood. Declarative and polar questions, which take up material from the problematic utterance, ask to confirm an hypothesis. Wh-questions, which are independent, reformulations or repetitions with additions (e.g., wh-substituted reprises) of the problematic utterance usually prompt for repetition, as do imperatives. Alternative questions prompt the addressee to disambiguate the hypothesis.

Answer: By definition, certain types of question prompt for certain answers. Therefore, the feature *answer* is closely linked to the sentence mood of the CR. As polar questions and declarative questions generally enquire about a proposition, i.e., an hypothesis or belief, they tend to receive yes/no answers, but repetitions are also possible. Wh-questions, alternative questions and imperatives tend to get answers providing additional information (i.e., reformulations and elaborations).

Extent: The *function* feature *extent* is logically independent from the *form* feature *completeness*, although they are strongly correlated. *Extent* is a binary feature indicating whether the CR points out a specific element or concerns the whole utterance. Most fragmentary declarative questions and fragmentary polar questions point out a specific element, especially when they are not independent but stand in some relation to the antecedent utterance. Independent complete imperatives address the whole previous utterance.

The correlations found in the Communicator Corpus are fairly consistent with those found in the Bielefeld Corpus, and thus we believe that the guidelines for generating CRs in task-oriented dialogues may be language independent, at least for German and English.

6 Summary and Future Work

In this paper we presented the results of a corpus study of naturally occurring CRs in task-oriented dialogue. Comparing our results to two other studies, one of a task-oriented corpus and one of a cor-

Form	Function			
	source	severity	extent	answer
mood	$\chi^2(24) = 112.20$ $p < 0.001$	$\chi^2(5) = 30.34$ $p < 0.001$	$\chi^2(5) = 24.25$ $df = p < 0.005$	$\chi^2(5) = 25.19$ $p < 0.001$
bound-tone	*indep.*	*indep.*	*indep.*	*indep.*
rel-antec	$\chi^2(24) = 108.23$ $p < 0.001$	$\chi^2(4) = 11.69$ $p < 0.005$	$\chi^2(4) = 42.58$ $p < 0.001$	*indep.*
complete	$\chi^2(7) = 27.39$ $p < 0.005$	*indep.*	$\chi^2(1) = 27.39$ $p < 0.001$	*indep.*

Table 7: Significance analysis for form/function correlations.

pus of everyday conversation, we found no significant differences in frequency of CRs and distribution of forms in the two task-oriented corpora, but many significant differences between CRs in task-oriented dialogue and everyday conversation. Our findings suggest that in task-oriented dialogues, humans use a cautious, but efficient strategy for clarification, preferring to present an hypothesis rather than ask the user to repeat or rephrase the problematic utterance. We also identified correlations between *function* and *form* features that can serve as a basis for generating more natural sounding CRs, which indicate a specific problem with understanding. In current work, we are studying data collected in a wizard-of-oz study in a multi-modal setting, in order to study clarification behavior in multi-modal dialogue.

Acknowledgements

The authors would like thank Kepa Rodriguez, Oliver Lemon, and David Reitter for help and discussion.

References

Christina L. Bennett and Alexander I. Rudnicky. 2002. The Carnegie Mellon Communicator Corpus. In *Proceedings of the International Conference of Spoken Language Processing (ICSLP02)*.

Lou Burnard. 2000. The British National Corpus Users Reference Guide. Technical report, Oxford Universiry Computing Services.

Herbert Clark. 1996. *Using Language*. Cambridge University Press.

Malte Gabsdil. 2003. Clarification in Spoken Dialogue Systems. *Proceedings of the 2003 AAAI Spring Symposium. Workshop on Natural Language Generation in Spoken and Written Dialogue*.

Jonathan Ginzburg and Robin Cooper. 2001. Resolving Ellipsis in Clarification. In *Proceedings of the 39th meeting of the Association for Computational Linguistics*.

Staffan Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, Goteborg University.

Tim Paek and Eric Horvitz. 2000. Conversation as Action Under Uncertainty. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*.

Tim Paek. 2003. Toward a Taxonomy of Communication Errors. In *ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*.

Matthew Purver, Jonathan Ginzburg, and Patrick Healey. 2003. On the Means for Clarification in Dialogue. In R. Smith and J. van Kuppevelt, editors, *Current and New Directions in Discourse and Dialogue*.

Matthew Purver. 2004. CLARIE: The Clarification Engine. In *Proceedings of the Eighth Workshop on Formal Semantics and Dialogue*.

Verena Rieser. 2004. Fragmentary Clarifications on Several Levels for Robust Dialogue Systems. Master's thesis, School of Informatics, University of Edinburgh.

Kepa J. Rodriguez and David Schlangen. 2004. Form, Intonation and Function of Clarification Requests in German Task-orientaded Spoken Dialogues. In *Proceedings of the Eighth Workshop on Formal Semantics and Dialogue*.

David Schlangen. 2004. Causes and Strategies for Request Clarification in Dialogue. *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*.

Gabriel Skantze. 2003. Exploring Human Error Handling Strategies: Implications for Spoken Dialogue Systems. In *ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*.

David R. Traum. 1999. Computational Models of Grounding in Collaborative Systems. In *Proceedings of the AAAI Fall Symposium on Psychological Models of Communication*.

Towards Finding and Fixing Fragments: Using ML to Identify Non-Sentential Utterances and their Antecedents in Multi-Party Dialogue

David Schlangen
Department of Linguistics
University of Potsdam
P.O. Box 601553
D-14415 Potsdam — Germany
das@ling.uni-potsdam.de

Abstract

Non-sentential utterances (e.g., short-answers as in “Who came to the party?”—“Peter.”) are pervasive in dialogue. As with other forms of ellipsis, the elided material is typically present in the context (e.g., the question that a short answer answers). We present a machine learning approach to the novel task of identifying fragments and their antecedents in multi-party dialogue. We compare the performance of several learning algorithms, using a mixture of structural and lexical features, and show that the task of identifying antecedents given a fragment can be learnt successfully ($f(0.5) = .76$); we discuss why the task of identifying fragments is harder ($f(0.5) = .41$) and finally report on a combined task ($f(0.5) = .38$).

1 Introduction

Non-sentential utterances (NSUs) as in (1) are pervasive in dialogue: recent studies put the proportion of such utterances at around 10% across different types of dialogue (Fernández and Ginzburg, 2002; Schlangen and Lascarides, 2003).

(1) a. A: Who came to the party?
B: Peter. (= *Peter came to the party.*)

b. A: I talked to Peter.
B: Peter Miller? (= *Was it Peter Miller you talked to?*)

c. A: Who was this? Peter Miller? (= *Was this Peter Miller?*

Such utterances pose an obvious problem for natural language processing applications, namely that the intended information (in (1-a)-B a proposition) has to be recovered from the uttered information (here, an NP meaning) with the help of information from the context.

While some systems that automatically resolve such fragments have recently been developed (Schlangen and Lascarides, 2002; Fernández et al., 2004a), they have the drawback that they require “deep” linguistic processing (full parses, and also information about discourse structure) and hence are not very robust. We have defined a well-defined subtask of this problem, namely identifying *fragments* (certain kinds of NSUs, see below) and their antecedents (in multi-party dialogue, in our case), and present a novel machine learning approach to it, which we hypothesise will be useful for tasks such as automatic meeting summarisation.[1]

The remainder of this paper is structured as follows. In the next section we further specify the task and different possible approaches to it. We then describe the corpus we used, some of its characteristics with respect to fragments, and the features we extracted from it for machine learning. Section 4 describes our experimental settings and reports the results. After a comparison to related work in Section 5, we close with a conclusion and some further

[1](Zechner and Lavie, 2001) describe a related task, linking questions and answers, and evaluate its usefulness in the context of automatic summarisation; see Section 5.

work that is planned.

2 The Tasks

As we said in the introduction, the main task we want to tackle is to align (certain kinds of) NSUs and their *antecedents*. Now, what characterises this kind of NSU, and what are their antecedents?

In the examples from the introduction, the NSUs can be resolved simply by looking at the previous utterance, which provides the material that is elided in them. In reality, however, the situation is not that simple, for three reasons: First, it is of course not always the *previous* utterance that provides this material (as illustrated by (2), where utterance 7 is resolved by utterance 1); in our data the average distance in fact is 2.5 utterances (see below).

(2)

1	B:	[. . .] What else should be done ?
2	C:	More intelligence .
3		More good intelligence .
4		Right .
5	D:	Intelligent intelligence .
6	B:	Better application of face and voice recognition .
7	C:	More [. . .] intermingling of the agencies , you know .

[from NSI 20011115]

Second, it's not even necessarily a single utterance that does this–it might very well be a span of utterances, or something that has to be inferred from such spans (parallel to the situation with pronouns, as discussed empirically e.g. in (Strube and Müller, 2003)). (3) shows an example where a new topic is broached by using an NSU. It is possible to analyse this as an answer to the *question under discussion* "what shall we organise for the party?", as (Fernández et al., 2004a) would do; a question, however, which is only *implicitly* posed by the previous discourse, and hence this is an example of an NSU that does not have an overt antecedent.

(3) [after discussing a number of different topics]

1	D:	So, equipment.
2		I can bring [. . .]

[from NSI 20011211]

Lastly, not all NSUs should be analysed as being the result of ellipsis: backchannels for example (like the "Right" in utterance 4 in (2) above) seem to directly fulfil their discourse function without any need for reconstruction.[2]

To keep matters simple, we concentrate in this paper on NSUs of a certain kind, namely those that a) do not predominantly have a discourse-management function (like for example backchannels), but rather convey messages (i.e., propositions, questions or requests)—this is what distinguishes *fragments* from other NSUs—and b) have individual utterances as antecedents. In the terminology of (Schlangen and Lascarides, 2003), fragments of the latter type are *resolution-via-identity*-fragments, where the elided information can be identified in the context and need not be inferred (as opposed to *resolution-via-inference*-fragments). Choosing only this special kind of NSUs poses the question whether this subgroup is distinguished from the general group of fragments by criteria that can be learnt; we will return to this below when we analyse the errors made by the classifier.

We have defined two approaches to this task. One is to split the task into two sub-tasks: identifying fragments in a corpus, and identifying antecedents for fragments. These steps are naturally performed sequentially to handle our main task, but they also allow the fragment classification decision to come from another source—a language-model used in an automatic speech recognition system, for example—and to use only the antecedent-classifier. The other approach is to do both at the same time, i.e. to classify pairs of utterances into those that combine a fragment and its antecedent and those that don't. We report the results of our experiments with these tasks below, after describing the data we used.

3 Corpus, Features, and Data Creation

3.1 Corpus

As material we have used six transcripts from the "NIST Meeting Room Pilot Corpus" (Garofolo et al., 2004), a corpus of recordings and transcriptions of multi-party meetings.[3] Those six transcripts con-

[2]The boundaries are fuzzy here, however, as backchannels can also be fragmental repetitions of previous material, and sometimes it is not clear how to classify a given utterance. A similar problem of classifying fragments is discussed in (Schlangen, 2003) and we will not go further into this here.

[3]We have chosen a multi-party setting because we are ultimately interested in automatic summarisation of meetings. In this paper here, however, we view our task as a "stand-alone task". Some of the problems resulting in the presence of many

average distance $\alpha - \beta$ (utterances):	2.5
α declarative	159 (52%)
α interrogative	140 (46%)
α unclassfd.	8 (2%)
β declarative	235 (76%)
β interrogative	(23%)
β unclassfd.	2 (0.7%)
α being last in their turn	142 (46%)
β being first in their turn	159 (52%)

Table 1: Some distributional characteristics. (α denotes antecedent, β fragment.)

sist of 5,999 utterances, among which we identified 307 fragment–antecedent pairs.[4,5] With 5.1% this is a lower rate than that reported for NSUs in other corpora (see above); but note that as explained above, we are actually only looking at a sub-class of all NSUs here.

For these pairs we also annotated some more attributes, which are summarised in Table 1. Note that the average distance is slightly higher than that reported in (Schlangen and Lascarides, 2003) for (2-party) dialogue (1.8); this is presumably due to the presence of more speakers who are able to reply to an utterance. Finally, we automatically annotated all utterances with part-of-speech tags, using `TreeTagger` (Schmid, 1994), which we've trained on the switchboard corpus of spoken language (Godfrey et al., 1992), because it contains, just like our corpus, speech disfluencies.[6]

We now describe the creation of the data we used for training. We first describe the data-sets for the different tasks, and then the features used to represent the events that are to be classified.

3.2 Data Sets

Data creation for the fragment-identification task (henceforth simply *fragment-task*) was straightforward: for each utterance, a number of features was derived automatically (see next section) and the correct class (fragment / other) was added. (Note that none of the manually annotated attributes were used.) This resulted in a file with 5,999 data points for classification. Given that there were 307 fragments, this means that in this data-set there is a ratio positives (fragments) vs. negatives (non-fragments) for the classifier of 1:20. To address this imbalance, we also ran the experiments with balanced data-sets with a ratio of 1:5.

The other tasks, antecedent-identification (*antecedent-task*) and antecedent-fragment-identification (*combined-task*) required the creation of data-sets containing pairs. For this we created an "accessibility window" going back from each utterance. Specifically, we included for each utterance a) all previous utterances of the same speaker from the same turn; and b) the three last utterances of every speaker, but only until one speaker took the turn again and up to a maximum of 6 previous utterances. To illustrate this method, given example (2) it would form pairs with utterance 7 as fragment-candidate and all of utterances 6–2, but not 1, because that violates condition b) (it is the second turn of speaker B).

In the case of (2), this exclusion would be a wrong decision, since 1 is in fact the antecedent for 7. In general, however, this dynamic method proved good at capturing as many antecedents as possible while keeping the number of data points manageable. It captured 269 antecedent-fragment pairs, which had an average distance of 1.84 utterances. The remaining 38 pairs which it missed had an average distance of 7.27 utterances, which means that to capture those we would have had to widen the window considerably. E.g., considering all previous 8 utterances would capture an additional 25 pairs, but at the cost of doubling the number of data points. We hence chose the approach described here, being aware of the introduction of a certain bias.

As we have said, we are trying to link *utterances*, one a fragment, the other its antecedent. The notion of *utterance* is however less well-defined than one might expect, and the segmentation of continuous speech into utterances is a veritable research problem on its own (see e.g. (Traum and Heeman, 1997)). Often it is arguable whether a prepositional

speakers are discussed below.

[4]We have used the MMAX tool (Müller and Strube, 2001)) for the annotation.

[5]To test the reliability of the annotation scheme, we had a subset of the data annotated by two annotators and found a satisfactory κ-agreement (Carletta, 1996) of $\kappa = 0.81$.

[6]The tagger is available free for academic research from `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html`.

Structural features	
`dis`	distance α – β, in utterances
`sspk`	same speaker yes/no
`nspk`	number speaker changes (= # turns)
`iqu`	number of intervening questions
`alt`	α last utterance in its turn?
`bft`	β first utterance in its turn?

Lexical / Utterance-based features	
`bvb`	(tensed) verb present in β?
`bds`	disfluency present in β?
`aqm`	α contains question mark
`awh`	α contains wh word
`bpr`	ratio of polar particles (*yes*, *no*, *maybe*, etc..) / other in β
`apr`	ratio of polar particles in α
`lal`	length of α
`lbe`	length of β
`nra`	ratio nouns / non-nouns in α
`nra`	ratio nouns / non-nouns in β
`rab`	ratio nouns in β that also occur in α
`rap`	ratio words in β that also occur in α
`god`	google similarity (see text)

Table 2: The Features

phrase for example should be analysed as an adjunct (and hence as not being an utterance on its own) or as a fragment. In our experiments, we have followed the decision made by the transcribers of the original corpus, since they had information (e.g. about pauses) which was not available to us.

For the antecedent-task, we include only pairs where β (the second utterance in the pair) is a fragment—since the task is to identify an antecedent for already identified fragments. This results in a data-set with 1318 data points (i.e., we created on average 4 pairs per fragment). This data-set is sufficiently balanced between positives and negatives, and so we did not create another version of it. The data for the combined-task, however, is much bigger, as it contains pairs for all utterances. It consists of 26,340 pairs, i.e. a ratio of roughly 1:90. For this reason we also used balanced data-sets for training, where the ratio was adjusted to 1:25.

3.3 Features

Table 2 lists the features we have used to represent the utterances. (In this table, and in this section, we denote the candidate for being a fragment with β and the candidate for being β's antecedent with α.)

We have defined a number of structural features, which give information about the (discourse-)structural relation between α and β. The rationale behind choosing them should be clear; `iqu` for example indicates in a weak way whether there might have been a topic change, and high `nspk` should presumably make an antecedent relation between α and β less likely.

We have also used some lexical or utterance-based features, which describe lexical properties of the individual utterances and lexical relations between them which could be relevant for the tasks. For example, the presence of a verb in β is presumably predictive for its being a fragment or not, as is the length. To capture a possible semantic relationship between the utterances, we defined two features. The more direct one, `rab`, looks at verbatim re-occurrences of nouns from α in β, which occur for example in check-questions as in (4) below.

(4) A: I saw Peter.
B: Peter? (= *Who is this Peter you saw?*)

Less direct semantic relations are intended to be captured by `god`, the second semantic feature we use.[7] It is computed as follows: for each pair (x, y) of nouns from α and β, Google is called (via the Google API) with a query for x, for y, and for x and y together. The similarity then is the average ratio of pair vs. individual term:

$$\textit{Google\_Similarity}(x,y) = (\frac{hits(x,y)}{hits(x)} + \frac{hits(x,y)}{hits(y)}) * \frac{1}{2}$$

We now describe the experiments we performed and their results.

4 Experiments and Results

4.1 Experimental Setup

For the learning experiments, we used three classifiers on all data-sets for the the three tasks:

• SLIPPER (Simple Learner with Iterative Pruning to Produce Error Reduction), (Cohen and Singer, 1999), which is a rule learner which combines the separate-and-conquer approach with confidence-rated boosting. It is unique among the classifiers that

[7]The name is short for *google distance*, which indicates its relatedness to the feature used by (Poesio et al., 2004); it is however a measure of *similarity*, not distance, as described above.

we have used in that it can make use of "set-valued" features, e.g. strings; we have run this learner both with only the features listed above and with the utterances (and POS-tags) as an additional feature.

• TIMBL (Tilburg Memory-Based Learner), (Daelemans et al., 2003), which implements a memory-based learning algorithm (IB1) which predicts the class of a test data point by looking at its distance to all examples from the training data, using some distance metric. In our experiments, we have used the weighted-overlap method, which assigns weights to all features.

• MAXENT, Zhang Le's C++ implementation[8] of maximum entropy modelling (Berger et al., 1996). In our experiments, we used L-BFGS parameter estimation.

We also implemented a naïve bayes classifier and ran it on the fragment-task, with a data-set consisting only of the strings and POS-tags.

To determine the contribution of all features, we used an iterative process similar to the one described in (Kohavi and John, 1997; Strube and Müller, 2003): we start with training a model using a baseline set of features, and then add each remaining feature individually, recording the gain (w.r.t. the f-measure ($f(0.5)$, to be precise)), and choosing the best-performing feature, incrementally until no further gain is recorded. All individual training- and evaluation-steps are performed using 8-fold cross-validation (given the small number of positive instances, more folds would have made the number of instances in the test set set too small).

The baselines were as follows: for the fragment-task, we used `bvb` and `lbe` as baseline, i.e. we let the classifier know the length of the candidate and whether the candidate contains a verb or not. For the antecedent-task we tested a very simple baseline, containing only of one feature, the distance between α and β (`dis`). The baseline for the combined-task, finally, was a combination of those two baselines, i.e. `bvb+lbe+dis`. The full feature-set for the fragment-task was `lbe, bvb, bpr, nrb, bft, bds` (since for this task there was no α to compute features of), for the two other tasks it was the complete set shown in Table 2.

[8]Available from `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`.

4.2 Results

The Tables 3–5 show the results of the experiments. The entries are roughly sorted by performance of the classifier used; for most of the classifiers and data-sets for each task we show the performance for baseline, intermediate feature set(s), and full feature-set, for the rest we only show the best-performing setting. We also indicate whether a balanced or unbalanced data set was used. I.e., the first three lines in Table 3 report on MaxEnt on a balanced data set for the fragment-task, giving results for the baseline, baseline+`nrb+bft`, and the full feature-set.

We begin with discussing the fragment task. As Table 3 shows, the three main classifiers perform roughly equivalently. Re-balancing the data, as expected, boosts recall at the cost of precision. For all settings (i.e., combinations of data-sets, feature-sets and classifier), except re-balanced maxent, the baseline (verb in β yes/no, and length of β) already has some success in identifying fragments, but adding the remaining features still boosts the performance. Having available the string (condition s.s; slipper with set valued features) interestingly does not help SLIPPER much.

Overall the performance on this task is not great. Why is that? An analysis of the errors made shows two problems. Among the false negatives, there is a high number of fragments like "yeah" and "mhm", which in their particular context were answers to questions, but that however occur much more often as backchannels (true negatives). The classifier, without having information about the context, can of course not distinguish between these cases, and goes for the majority decision. Among the false positives, we find utterances that are indeed non-sentential, but for which no antecedent was marked (as in (3) above), i.e., which are not fragments in our narrow sense. It seems, thus, that the required distinctions are not ones that can be reliably learnt from looking at the fragments alone.

The antecedent-task was handled more satisfactorily, as Table 4 shows. For this task, a naïve baseline ("always take previous utterance") preforms relatively well already; however, all classifiers were able to improve on this, with a slight advantage for the maxent model ($f(0.5) = 0.76$). As the entry for MaxEnt shows, adding to the baseline-features

Data Set	Cl.	Recall	Precision	$\mathbf{f(0.5)}$	$\mathbf{f(1.0)}$	$\mathbf{f(2.0)}$
B; `bl`	m	0.00	0.00	0.00	0.00	0.00
B; `bl+nrb+bft`	m	36.39	31.16	0.31	0.33	0.35
B; `all`	m	**40.61**	**44.10**	**0.43**	**0.42**	**0.41**
UB; `all`	m	**22.13**	**65.06**	**0.47**	**0.33**	**0.25**
B; `bl`	t	31.77	21.20	0.22	0.24	0.28
B; `bl+nrb+bpr+bds`	t	**42.18**	**41.26**	**0.41**	**0.42**	**0.42**
B; `all`	t	44.54	32.74	0.34	0.37	0.41
UB; `bl+nrb`	t	**26.22**	**59.05**	**0.47**	**0.36**	**0.29**
B; `bl`	s	21.07	16.95	0.17	0.18	0.20
B; `bl+nrb+bft+bds`	s	**36.37**	**49.28**	**0.46**	**0.41**	**0.38**
B; `all`	s	36.67	43.31	0.42	0.40	0.38
UB; `bl+nrb`	s	**28.28**	**57.88**	**0.48**	**0.38**	**0.31**
B	s.s	32.57	42.96	0.40	0.36	0.34
B	b	55.62	19.75	0.23	0.29	0.41
UB	b	66.50	20.00	0.23	0.31	0.45

Table 3: Results for the fragment task. (Cl. = classifier used, where s = slipper, s.s = slipper + set-valued features, t = timbl, m = maxent, b = naive bayes; UB/B = (un)balanced training data.)

Data Set	Cl.	Recall	Precision	$\mathbf{f(0.5)}$	$\mathbf{f(1.0)}$	$\mathbf{f(2.0)}$
dis=1	-	44.95	44.81	0.45	0.45	0.45
UB; `bl`	m	0	0	0.0	0.0	0.0
UB; `bl+awh`	m	43.21	52.90	0.50	0.47	0.45
UB; `bl+awh+god`	m	36.98	75.31	0.62	0.50	0.41
UB; `bl+awh+god+lbe+lal+iqu+nra+buh`	m	**64.26**	**80.39**	**0.76**	**0.71**	**0.67**
UB; `all`	m	58.16	73.57	0.69	0.64	0.60
UB; `bl`	s	0.00	0.00	0.00	0.00	0.00
UB; `bl+aqm`	s	36.65	78.44	0.63	0.49	0.41
UB; `bl+aqm+rab+iqu+lal`	s	**49.72**	**79.75**	**0.71**	**0.61**	**0.54**
UB; `all`	s	49.43	72.57	0.66	0.58	0.52
UB; `bl`	t	0	0	0.0	0.0	0.0
UB; `bl+aqm`	t	36.98	73.58	0.61	0.49	0.41
UB; `bl+aqm+awh+rab+iqu`	t	**46.41**	**77.65**	**0.68**	**0.58**	**0.50**
UB; `all`	t	60.57	58.74	0.59	0.60	0.60

Table 4: Results for the antecedent task.

Data Set	Cl.	Recall	Precision	$\mathbf{f(0.5)}$	$\mathbf{f(1.0)}$	$\mathbf{f(2.0)}$
B; `bl`	m	0.00	0.00	0.00	0.00	0.00
B; `bl+rap`	m	5.83	40.91	0.18	0.10	0.07
B; `bl+rap+god`	m	7.95	55.83	0.25	0.14	0.10
B; `bl+rap+god+nspk`	m	11.70	49.15	0.30	0.19	0.14
B; `bl+rap+god+nspk+alt+awh+nra+lal`	m	**20.27**	**50.02**	**0.38**	**0.28**	**0.23**
B; `all`	m	23.29	43.79	0.36	0.30	0.25
UB; `bl+rap+god+nspk+iqu+nra+bds+rab+awh`	m	**13.01**	**54.87**	**0.33**	**0.21**	**0.15**
B; `bl`	s	0.00	0.00	0.00	0.00	0.00
B; `bl+god`	s	11.80	35.60	0.25	0.17	0.13
B; `bl+god+bds`	s	14.44	46.98	0.32	0.22	0.17
B; `all`	s	**17.78**	**41.96**	**0.32**	**0.24**	**0.20**
UB; `bl+alt+bds+god+sspk+rap`	s	**11.37**	**56.34**	**0.31**	**0.19**	**0.13**
B; `bl`	t	0.00	0.00	0.00	0.00	0.00
B; `bl+god`	t	**17.20**	**29.09**	**0.25**	**0.21**	**0.19**
B; `all`	t	17.87	19.97	0.19	0.19	0.18
UB; `bl+god+iqu+rab`	t	**14.24**	**41.63**	**0.29**	**0.21**	**0.16**
B; `bl+rab+buh`	s.s	**8.63**	**54.20**	**0.26**	**0.15**	**0.10**

Table 5: Results for the combined task.

information about whether α is a question or not already boost the performance considerably. An analysis of the predictions of this model then indeed shows that it already captures cases of question and answer pairs quite well. Adding the similarity feature `god` then gives the model information about semantic relatedness, which, as hypothesised, captures elaboration-type relations (as in (1-b) and (1-c) above). Structural information (`iqu`) further improves the model; however, the remaining features only seem to add interfering information, for performance using the full feature-set is worse.

If one of the problems of the fragment-task was that information about the context is required to distinguish fragments and backchannels, then the hope could be that in the combined-task the classifier would able to capture these cases. However, the performance of all classifiers on this task is not satisfactory, as Table 5 shows; in fact, it is even slightly worse than the performance on the fragment task alone. We speculate that instead of of cancelling out mistakes in the other part of the task, the two goals (let β be a fragment, and α a typical antecedent) interfere during optimisation of the rules.

To summarise, we have shown that the task of identifying the antecedent of a given fragment is learnable, using a feature-set that combines structural and lexical features; in particular, the inclusion of a measure of semantic relatedness, which was computed via queries to an internet search engine, proved helpful. The task of identifying (*resolution-via-identity*) fragments, however, is hindered by the high number of non-sentential utterances which can be confused with the kinds of fragments we are interested in. Here it could be helpful to have a method that identifies and filters out backchannels, presumably using a much more local mechanism (as for example proposed in (Traum, 1994)). Similarly, the performance on the combined task is low, also due to a high number of confusions of backchannels and fragments. We discuss an alternative set-up below.

5 Related Work

To our knowledge, the tasks presented here have so far not been studied with a machine learning approach. The closest to our problem is (Fernández et al., 2004b), which discusses *classifying* certain types of fragments, namely questions of the type "Who?", "When?", etc. (*sluices*). However, that paper does not address the task of *identifying* those in a corpus (which in any case should be easier than our fragment-task, since those fragments cannot be confused with backchannels).

Overlapping from another direction is the work presented in (Zechner and Lavie, 2001), where the task of aligning questions and answers is tackled. This subsumes the task of identifying question-antecedents for short-answers, but again is presumably somewhat simpler than our general task, because questions are easier to identify. The authors also evaluate the use of the alignment of questions and answers in a summarisation system, and report an increase in summary fluency, without a compromise in informativeness. This is something we hope to be able to show for our tasks as well.

There are also similarities, especially of the antecedent task, to the pronoun resolution task (see e.g. (Strube and Müller, 2003; Poesio et al., 2004)). Interestingly, our results for the antecedent task are close to those reported for that task. The problem of identifying the units in need of an antecedent, however, is harder for us, due to the problem of there being a large number of non-sentential utterances that cannot be linked to a single utterance as antecedent. In general, this seems to be the main difference between our task and the ones mentioned here, which concentrate on more easily identified markables (questions, sluices, and pronouns).

6 Conclusions and Further Work

We have presented a machine learning approach to the task of identifying fragments and their antecedents in multi-party dialogue. This represents a well-defined subtask of computing discourse structure, which to our knowledge has not been studied so far. We have shown that the task of identifying the antecedent of a given fragment is learnable, using features that provide information about the structure of the discourse between antecedent and fragment, and about semantic closeness.

The other tasks, identifying fragments and the combined tasks, however, did not perform as well, mainly because of a high rate of confusions between general non-sentential utterances and fragments (in our sense). In future work, we will try a modified approach, where the detection of fragments is integrated with a classification of utterances as backchannels, fragments, or full sentences, and where the antecedent task only ranks pairs, leaving open the possibility of excluding a supposed fragment by using contextual information. Lastly, we are planning to integrate our classifier into a processing pipeline after the pronoun resolution step, to see whether this would improve both our performance and the quality of automatic meeting summarisations.[9]

References

Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2):249–254.

William Cohen and Yoram Singer. 1999. A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, Orlando, Florida, July. AAAI.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2003. TiMBL: Tilburg memory based learner, version 5.0, reference guide. ILC Technical Report 03-10, Induction of Linguistic Knowledge; Tilburg University. Available from `http://ilk.uvt.nl/downloads/pub/...papers/ilk0310.pdf`.

Raquel Fernández and Jonathan Ginzburg. 2002. Non-sentential utterances in dialogue: A corpus-based study. In Kristiina Jokinen and Susan McRoy, editors, *Proceedings of the Third SIGdial Workshop on Discourse and Dialogue*, pages 15–26, Philadelphia, USA, July. ACL Special Interest Group on Dialog.

Raquel Fernández, Jonathan Ginzburg, Howard Gregory, and Shalom Lappin. 2004a. Shards: Fragment resolution in dialogue. In H. Bunt and R. Muskens, editors, *Computing Meaning*, volume 3. Kluwer.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2004b. Classifying ellipsis in dialogue: A machine learning approach. In *Proceedings of COLING 2004*, Geneva, Switzerland, August.

John S. Garofolo, Christophe D. Laprun, Martial Michel, Vincent M. Stanford, and Elham Tabassi. 2004. The NITS meeting room pilot corpus. In *Proceedings of the International Language Resources Conference (LREC04)*, Lisbon, Portugal, May.

J.J. Godfrey, E. C. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and devlopment. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 517–520, San Francisco, USA, March.

Ron Kohavi and George H. John. 1997. Wrappers for feature selection. *Artificial Intelligence Journal*, 97(1–2):273–324.

Christoph Müller and Michael Strube. 2001. MMAX: A Tool for the Annotation of Multi-modal Corpora. In *Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 45–50, Seattle, USA, August.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of the 42nd annual meeting of the Association for Computational Linguistics*, pages 144–151, Barcelona, Spain, July.

David Schlangen and Alex Lascarides. 2002. Resolving fragments using discourse information. In Johan Bos, Mary Ellen Foster, and Colin Matheson, editors, *Proceedings of the 6th International Workshop on Formal Semantics and Pragmatics of Dialogue (EDILOG 2002)*, pages 161–168, Edinburgh, September.

David Schlangen and Alex Lascarides. 2003. The interpretation of non-sentential utterances in dialogue. In Alexander Rudnicky, editor, *Proceedings of the 4th SIGdial workshop on Discourse and Dialogue*, Sapporo, Japan, July.

David Schlangen. 2003. *A Coherence-Based Approach to the Interpretation of Non-Sentential Utterances in Dialogue*. Ph.D. thesis, School of Informatics, University of Edinburgh, Edinburgh, UK.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Michael Strube and Christoph Müller. 2003. A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Lingustics*, Sapporo, Japan.

D. Traum and P. Heeman. 1997. Utterance units in spoken dialogue. In E. Maier, M. Mast, and S. LuperFoy, editors, *Dialogue Processing in Spoken Language Systems*, Lecture Notes in Artificial Intelligence. Springer-Verlag.

David R. Traum. 1994. *A Computational Theory of Grounding in Natural Language Conversation*. Ph.D. thesis, Computer Science, University of Rochester, Rochester, USA, December.

Klaus Zechner and Anton Lavie. 2001. Increasing the coherence of spoken dialogue summaries by cross-speaker information linking. In *Proceedings of the NAAACL Workshop on Automatic Summarisation*, Pittsburgh, USA, June.

[9]**Acknowledgements:** We would like to acknowledge helpful discussions with Jason Baldridge and Michael Strube during the early stages of the project, and helpful comments from the anonymous reviewers.

Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases

Chris Callison-Burch **Colin Bannard**
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW
{chris,colin}@linearb.co.uk

Josh Schroeder
Linear B Ltd.
39 B Cumberland Street
Edinburgh EH3 6RA
josh@linearb.co.uk

Abstract

In this paper we describe a novel data structure for phrase-based statistical machine translation which allows for the retrieval of arbitrarily long phrases while simultaneously using less memory than is required by current decoder implementations. We detail the computational complexity and average retrieval times for looking up phrase translations in our suffix array-based data structure. We show how sampling can be used to reduce the retrieval time by orders of magnitude with no loss in translation quality.

1 Introduction

Statistical machine translation (SMT) has an advantage over many other statistical natural language processing applications in that training data is regularly produced by other human activity. For some language pairs very large sets of training data are now available. The publications of the European Union and United Nations provide gigbytes of data between various language pairs which can be easily mined using a web crawler. The Linguistics Data Consortium provides an excellent set of off the shelf Arabic-English and Chinese-English parallel corpora for the annual NIST machine translation evaluation exercises.

The size of the NIST training data presents a problem for phrase-based statistical machine translation. Decoders such as Pharaoh (Koehn, 2004) primarily use lookup tables for the storage of phrases and their translations. Since retrieving longer segments of human translated text generally leads to better translation quality, participants in the evaluation exercise try to maximize the length of phrases that are stored in lookup tables. The combination of large corpora and long phrases means that the table size can quickly become unwieldy.

A number of groups in the 2004 evaluation exercise indicated problems dealing with the data. Coping strategies included limiting the length of phrases to something small, not using the entire training data set, computing phrases probabilities on disk, and filtering the phrase table down to a manageable size after the testing set was distributed. We present a data structure that is easily capable of handling the largest data sets currently available, and show that it can be scaled to much larger data sets.

In this paper we:

- Motivate the problem with storing enumerated phrases in a table by examining the memory requirements of the method for the NIST data set
- Detail the advantages of using long phrases in SMT, and examine their potential coverage
- Describe a suffix array-based data structure which allows for the retrieval of translations of arbitrarily long phrases, and show that it requires far less memory than a table
- Calculate the computational complexity and average time for retrieving phrases and show how this can be sped up by orders of magnitude with no loss in translation accuracy

2 Related Work

Koehn et al. (2003) compare a number of different approaches to phrase-based statistical machine

Proceedings of the 43rd Annual Meeting of the ACL, pages 255–262,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

length	num uniq (mil)	average # translations	avg trans length
1	.88	8.322	1.37
2	16.5	1.733	2.35
3	42.6	1.182	3.44
4	58.7	1.065	4.58
5	65.0	1.035	5.75
6	66.4	1.022	6.91
7	65.8	1.015	8.07
8	64.3	1.012	9.23
9	62.2	1.010	10.4
10	59.9	1.010	11.6

Table 1: Statistics about Arabic phrases in the NIST-2004 large data track.

length	entries (mil)	words (mil)	memory (gigs)	including alignments
1	7.3	10	.1	.11
2	36	111	.68	.82
3	86	412	2.18	2.64
4	149	933	4.59	5.59
5	216	1,645	7.74	9.46
6	284	2,513	11.48	14.07
7	351	3,513	15.70	19.30
8	416	4,628	20.34	25.05
9	479	5,841	25.33	31.26
10	539	7,140	30.62	37.85

Table 2: Estimated size of lookup tables for the NIST-2004 Arabic-English data

translation including the joint probability phrase-based model (Marcu and Wong, 2002) and a variant on the alignment template approach (Och and Ney, 2004), and contrast them to the performance of the word-based IBM Model 4 (Brown et al., 1993). Most relevant for the work presented in this paper, they compare the effect on translation quality of using various lengths of phrases, and the size of the resulting phrase probability tables.

Tillmann (2003) further examines the relationship between maximum phrase length, size of the translation table, and accuracy of translation when inducing block-based phrases from word-level alignments. Venugopal et al. (2003) and Vogel et al. (2003) present methods for achieving better translation quality by growing incrementally larger phrases by combining smaller phrases with overlapping segments.

3 Scaling to Long Phrases

Table 1 gives statistics about the Arabic-English parallel corpus used in the NIST large data track. The corpus contains 3.75 million sentence pairs, and has 127 million words in English, and 106 million words in Arabic. The table shows the number of unique Arabic phrases, and gives the average number of translations into English and their average length.

Table 2 gives estimates of the size of the lookup tables needed to store phrases of various lengths, based on the statistics in Table 1. The number of unique entries is calculated as the number unique phrases times the average number of translations. The number of words in the table is calculated as the number of unique phrases times the phrase length plus the number of entries times the average translation length. The memory is calculated assuming that each word is represented with a 4 byte integer, that each entry stores its probability as an 8 byte double and that each word alignment is stored as a 2 byte short. Note that the size of the table will vary depending on the phrase extraction technique.

length	coverage	length	coverage
1	93.5%	6	4.70%
2	73.3%	7	2.95%
3	37.1%	8	2.14%
4	15.5%	9	1.99%
5	8.05%	10	1.49%

Table 3: Lengths of phrases from the training data that occur in the NIST-2004 test set

Table 3 gives the percent of the 35,313 word long test set which can be covered using only phrases of the specified length or greater. The table shows the efficacy of using phrases of different lengths. The table shows that while the rate of falloff is rapid, there are still multiple matches of phrases of length 10. The longest matching phrase was one of length 18. There is little generalization in current SMT implementations, and consequently longer phrases generally lead to better translation quality.

3.1 Why use phrases?

Statistical machine translation made considerable advances in translation quality with the introduction of phrase-based translation. By increasing the size of the basic unit of translation, phrase-based machine translation does away with many of the problems associated with the original word-based formulation of statistical machine translation (Brown et al., 1993), in particular:

- The Brown et al. (1993) formulation doesn't have a direct way of translating phrases; instead they specify a *fertility* parameter which is used to replicate words and translate them individually.

- With units as small as words, a lot of reordering has to happen between languages with different word orders. But the *distortion* parameter is a poor explanation of word order.

Phrase-based SMT overcomes the first of these problems by eliminating the fertility parameter and directly handling word-to-phrase and phrase-to-phrase mappings. The second problem is alleviated through the use of multi-word units which reduce the dependency on the distortion parameter. Less word re-ordering need occur since local dependencies are frequently captured. For example, common adjective-noun alternations are memorized. However, since this linguistic information is not encoded in the model, unseen adjective noun pairs may still be handled incorrectly.

By increasing the length of phrases beyond a few words, we might hope to capture additional non-local linguistic phenomena. For example, by memorizing longer phrases we may correctly learn case information for nouns commonly selected by frequently occurring verbs; we may properly handle discontinuous phrases (such as French negation, some German verb forms, and English verb particle constructions) that are neglected by current phrase-based models; and we may by chance capture some agreement information in coordinated structures.

3.2 Deciding what length of phrase to store

Despite the potential gains from memorizing longer phrases, the fact remains that as phrases get longer there is a decreasing likelihood that they will be repeated. Because of the amount of memory required to store a phrase table, in current implementations a choice is made as to the maximum length of phrase to store.

length	coverage	length	coverage
1	96.3%	6	21.9%
2	94.9%	7	11.2%
3	86.1%	8	6.16%
4	65.6%	9	3.95%
5	40.9%	10	2.90%

Table 4: Coverage using only repeated phrases of the specified length

Based on their analysis of the relationship between translation quality and phrase length, Koehn et al. (2003) suggest limiting phrase length to three words or less. This is entirely a practical suggestion for keeping the phrase table to a reasonable size, since they measure minor but incremental improvement in translation quality up to their maximum tested phrase length of seven words.[1]

Table 4 gives statistics about phrases which occur more than once in the English section of the Europarl corpus (Koehn, 2002) which was used in the Koehn et al. (2003) experiments. It shows that the percentage of words in the corpus that can be covered by repeated phrases falls off rapidly at length 6, but that even phrases up to length 10 are able to cover a non-trivial portion of the corpus. This draws into question the desirability of limiting phrase retrieval to length three.

The decision concerning what length of phrases to store in the phrase table seems to boil down to a practical consideration: one must weigh the likelihood of retrieval against the memory needed to store longer phrases. We present a data structure where this is not a consideration. Our suffix array-based data structure allows the retrieval of arbitrarily long phrases, while simultaneously requiring far less memory than the standard table-based representation.

[1]While the improvements to translation quality reported in Koehn et al. (2003) are minor, their evaluation metric may not have been especially sensitive to adding longer phrases. They used the Bleu evaluation metric (Papineni et al., 2002), but capped the n-gram precision at 4-grams.

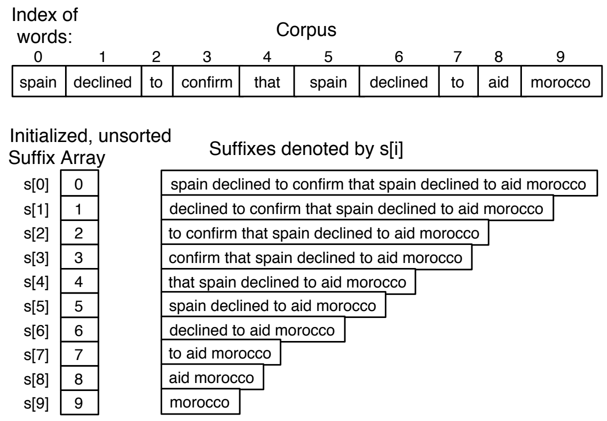

Index of words:
Corpus
0 1 2 3 4 5 6 7 8 9
spain declined to confirm that spain declined to aid morocco
Initialized, unsorted Suffix Array
Suffixes denoted by s[i]
s[0] 0 spain declined to confirm that spain declined to aid morocco
s[1] 1 declined to confirm that spain declined to aid morocco
s[2] 2 to confirm that spain declined to aid morocco
s[3] 3 confirm that spain declined to aid morocco
s[4] 4 that spain declined to aid morocco
s[5] 5 spain declined to aid morocco
s[6] 6 declined to aid morocco
s[7] 7 to aid morocco
s[8] 8 aid morocco
s[9] 9 morocco

Figure 1: An initialized, unsorted suffix array for a very small corpus

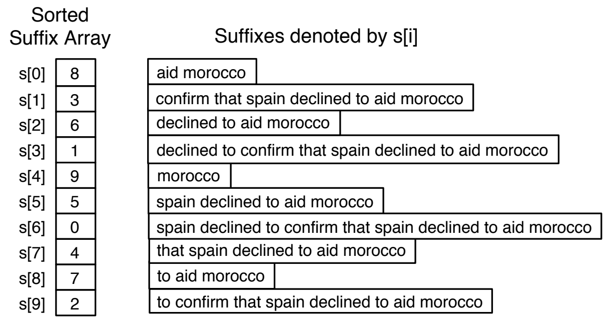

Sorted Suffix Array
Suffixes denoted by s[i]
s[0] 8 aid morocco
s[1] 3 confirm that spain declined to aid morocco
s[2] 6 declined to aid morocco
s[3] 1 declined to confirm that spain declined to aid morocco
s[4] 9 morocco
s[5] 5 spain declined to aid morocco
s[6] 0 spain declined to confirm that spain declined to aid morocco
s[7] 4 that spain declined to aid morocco
s[8] 7 to aid morocco
s[9] 2 to confirm that spain declined to aid morocco

Figure 2: A sorted suffix array and its corresponding suffixes

4 Suffix Arrays

The suffix array data structure (Manber and Myers, 1990) was introduced as a space-economical way of creating an index for string searches. The suffix array data structure makes it convenient to compute the frequency and location of any substring or n-gram in a large corpus. Abstractly, a suffix array is an alphabetically-sorted list of all suffixes in a corpus, where a suffix is a substring running from each position in the text to the end. However, rather than actually storing all suffixes, a suffix array can be constructed by creating a list of references to each of the suffixes in a corpus. Figure 1 shows how a suffix array is initialized for a corpus with one sentence. Each index of a word in the corpus has a corresponding place in the suffix array, which is identical in length to the corpus. Figure 2 shows the final state of the suffix array, which is as a list of the indices of words in the corpus that corresponds to an alphabetically sorted list of the suffixes.

The advantages of this representation are that it is compact and easily searchable. The total size of the suffix array is a constant amount of memory. Typically it is stored as an array of integers where the array is the same length as the corpus. Because it is organized alphabetically, any phrase can be quickly located within it using a binary search algorithm.

Yamamoto and Church (2001) show how to use suffix arrays to calculate a number of statistics that are interesting in natural language processing applications. They demonstrate how to calculate term frequency / inverse document frequency (*tf / idf*) for all n-grams in very large corpora, as well as how to use these frequencies to calculate n-grams with high mutual information and residual inverse document frequency. Here we show how to apply suffix arrays to parallel corpora to calculate phrase translation probabilities.

4.1 Applied to parallel corpora

In order to adapt suffix arrays to be useful for statistical machine translation we need a data structure with the following elements:

- A suffix array created from the source language portion of the corpus, and another created from the target language portion of the corpus,
- An index that tells us the correspondence between sentence numbers and positions in the source and target language corpora,
- An alignment $\mathbf{a}$ for each sentence pair in the parallel corpus, where $\mathbf{a}$ is defined as a subset of the Cartesian product of the word positions in a sentence $\mathbf{e}$ of length I and a sentence $\mathbf{f}$ of length J:

$$\mathbf{a} \subseteq \{(i, j) : i = 1...I; j = 1...J\}$$

- A method for extracting the translationally equivalent phrase for a subphrase given an aligned sentence pair containing that subphrase.

The total memory usage of the data structure is thus the size of the source and target corpora, plus the size of the suffix arrays (identical in length to the corpora), plus the size of the two indexes that correlate sentence positions with word positions, plus the size of the alignments. Assuming we use *ints* to represent words and indices, and *shorts* to represent word alignments, we get the following memory usage:

$$2 * \text{num words in source corpus} * sizeof(int) +$$

$$2 * \text{num words in target corpus} * sizeof(int) +$$

$$2 * \text{number sentence pairs} * sizeof(int) +$$

$$\text{number of word alignments} * sizeof(short)$$

The total amount of memory required to store the NIST Arabic-English data using this data structure is

$$2 * 105{,}994{,}774 * sizeof(int) +$$

$$2 * 127{,}450{,}473 * sizeof(int) +$$

$$2 * 3{,}758{,}904 * sizeof(int) +$$

$$92{,}975{,}229 * sizeof(short)$$

Or just over 2 Gigabytes.

4.2 Calculating phrase translation probabilities

In order to produce a set of phrase translation probabilities, we need to examine the ways in which they are calculated. We consider two common ways of calculating the translation probability: using the maximum likelihood estimator (MLE) and smoothing the MLE using lexical weighting.

The maximum likelihood estimator for the probability of a phrase is defined as

$$p(\bar{f}|\bar{e}) = \frac{count(\bar{f}, \bar{e})}{\sum_{\bar{f}} count(\bar{f}, \bar{e})} \qquad (1)$$

Where $count(\bar{f}, \bar{e})$ gives the total number of times the phrase $\bar{f}$ was aligned with the phrase $\bar{e}$ in the parallel corpus. We define phrase alignments as follows. A substring $\bar{e}$ consisting of the words at positions $l...m$ is aligned with the phrase $\bar{f}$ by way of the subalignment

$$\mathbf{s} = \mathbf{a} \cap \{(i,j) : i = l...m, j = 1...J\}$$

The aligned phrase $\bar{f}$ is the subphrase in $\mathbf{f}$ which spans from $min(j)$ to $max(j)$ for $j|(i,j) \in \mathbf{s}$.

The procedure for generating the counts that are used to calculate the MLE probability using our suffix array-based data structures is:

1. Locate all the suffixes in the English suffix array which begin with the phrase $\bar{e}$. Since the suffix array is sorted alphabetically we can easily find the first occurrence $s[k]$ and the last occurrence $s[l]$. The length of the span in the suffix array $l-k+1$ indicates the number of occurrences of $\bar{e}$ in the corpus. Thus the denominator $\sum_{\bar{f}} count(\bar{f}, \bar{e})$ can be calculated as $l - k + 1$.

2. For each of the matching phrases $s[i]$ in the span $s[k]...s[l]$, look up the value of $s[i]$ which is the word index w of the suffix in the English corpus. Look up the sentence number that includes w, and retrieve the corresponding sentences $\mathbf{e}$ and $\mathbf{f}$, and their alignment $\mathbf{a}$.

3. Use $\mathbf{a}$ to extract the target phrase $\bar{f}$ that aligns with the phrase $\bar{e}$ that we are searching for. Increment the count for $< \bar{f}, \bar{e} >$.

4. Calculate the probability for each unique matching phrase $\bar{f}$ using the formula in Equation 1.

A common alternative formulation of the phrase translation probability is to lexically weight it as follows:

$$p_{lw}(\bar{f}|\bar{e}, \mathbf{s}) = \prod_{i=1}^{n} \frac{1}{|\{i|(i,j) \in \mathbf{s}\}|} \sum_{\forall (i,j) \in \mathbf{s}} p(f_j|e_i) \qquad (2)$$

Where n is the length of $\bar{e}$.

In order to use lexical weighting we would need to repeat steps 1-4 above for each word e_i in $\bar{e}$. This would give us the values for $p(f_j|e_i)$. We would further need to retain the subphrase alignment $\mathbf{s}$ in order to know the correspondence between the words $(i,j) \in \mathbf{s}$ in the aligned phrases, and the total number of foreign words that each e_i is aligned with ($|\{i|(i,j) \in \mathbf{s}\}|$). Since a phrase alignment $< \bar{f}, \bar{e} >$ may have multiple possible word-level alignments, we retain a set of alignments S and take the maximum:

$$p(\bar{f}|\bar{e}, S) = p(\bar{f}|\bar{e}) * \arg\max_{\mathbf{s} \in S} p_{lw}(\bar{f}|\bar{e}, \mathbf{s}) \qquad (3)$$

Thus our suffix array-based data structure can be used straightforwardly to look up all aligned translations for a given phrase and calculate the probabilities on-the-fly. In the next section we turn to the computational complexity of constructing phrase translation probabilities in this way.

5 Computational Complexity

Computational complexity is relevant because there is a speed-memory tradeoff when adopting our data structure. What we gained in memory efficiency may be rendered useless if the time it takes to calculate phrase translation probabilities is unreasonably long. The computational complexity of looking up items in a hash table, as is done in current table-based data structures, is extremely fast. Looking up a single phrase can be done in unit time, $O(1)$.

The computational complexity of our method has the following components:

- The complexity of finding all occurrences of the phrase in the suffix array
- The complexity of retrieving the associated aligned sentence pairs given the positions of the phrase in the corpus
- The complexity of extracting all aligned phrases using our phrase extraction algorithm
- The complexity of calculating the probabilities given the aligned phrases

The methods we use to execute each of these, and their complexities are as follow:

- Since the array is sorted, finding all occurrences of the English phrase is extremely fast. We can do two binary searches: one to find the first occurrence of the phrase and a second to find the last. The computational complexity is therefore bounded by $O(2\log(n))$ where n is the length of the corpus.
- We use a similar method to look up the sentences $\mathbf{e_i}$ and $\mathbf{f_i}$ and word-level alignment $\mathbf{a_i}$ that are associated with the position w_i in the corpus of each phrase occurrence $\bar{e}_i$. The complexity is $O(k * 2\log(m))$ where k is the number of occurrences of $\bar{e}$ and m is the number of sentence pairs in the parallel corpus.
- The complexity of extracting the aligned phrase for a single occurrence of $\bar{e}_i$ is $O(2\log(|\mathbf{a_i}|)$ to get the subphrase alignment $\mathbf{s_i}$, since we store the alignments in a sorted array. The complexity of then getting $\bar{f}_i$ from $\mathbf{s_i}$ is $O(length(\bar{f}_i))$.
- The complexity of summing over all aligned phrases and simultaneously calculating their probabilities is $O(k)$.

Thus we have a total complexity of:

$$O(2\log(n) + k * 2\log(m) \qquad (4)$$

$$+ \sum_{\mathbf{a_i}, \bar{f}_i | \bar{e}_i}^{\bar{e}_1 \ldots \bar{e}_k} (2\log(|\mathbf{a_i}|) + length(\bar{f}_i)) + k) \qquad (5)$$

for the MLE estimation of the translation probabilities for a single phrase. The complexity is dominated by the k terms in the equation, when the number of occurrences of the phrase in the corpus is high. Phrases with high frequency may cause excessively long retrieval time. This problem is exacerbated when we shift to a lexically weighted calculation of the phrase translation probability. The complexity will be multiplied across each of the component words in the phrase, and the component words themselves will be more frequent than the phrase.

phrase	freq	O	time (ms)
respect for the dead	3	80	24
since the end of the cold war	19	240	136
the parliament	1291	4391	1117
of the	290921	682550	218369

Table 5: Examples of O and calculation times for phrases of different frequencies

Table 5 shows example times for calculating the translation probabilities for a number of phrases. For frequent phrases like *of the* these times get unacceptably long. While our data structure is perfect for overcoming the problems associated with storing the translations of long, infrequently occurring phrases, it in a way introduces the converse problem. It has a clear disadvantage in the amount of time it takes to retrieve commonly occurring phrases. In the next section we examine the use of sampling to speed up the calculation of translation probabilities for very frequent phrases.

6 Sampling

Rather than compute the phrase translation probabilities by examining the hundreds of thousands of occurrences of common phrases, we instead sample from a small subset of the occurrences. It is unlikely that we need to extract the translations of all occurrences of a high frequency phrase in order to get a good approximation of their probabilities. We instead cap the number of occurrences that we consider, and thus give a maximum bound on k in Equation 5.

In order to determine the effect of different levels of sampling, we compare the translation quality against cumulative retrieval time for calculating the phrase translation probabilities for all subphrases in an evaluation set. We translated a held out set of 430 German sentences with 50 words or less into English. The test sentences were drawn from the 01/17/00 proceedings of the Europarl corpus. The remainder of the corpus (1 million sentences) was used as training data to calculate the phrase translation probabilities. We calculated the translation quality using Bleu's modified n-gram precision metric (Papineni et al., 2002) for n-grams of up to length four. The framework that we used to calculate the translation probabilities was similar to that detailed in Koehn et al. (2003). That is:

$$\begin{aligned} \hat{\mathrm{e}} &= \arg\max_{\mathbf{e_1^I}} p(\mathbf{e_1^I}|\mathbf{f_1^I}) & (6)\\ &= \arg\max_{\mathbf{e_1^I}} p_{LM}(\mathbf{e_1^I}) * & (7)\\ & \prod_{i=1}^{I} p(\bar{f}_i|\bar{e}_i) d(a_i - b_{i-1}) p_{lw}(\bar{f}_i|\bar{e}_i, \mathbf{a}) & (8)\end{aligned}$$

Where p_{LM} is a language model probability and d is a distortion probability which penalizes movement.

Table 6 gives a comparison of the translation quality under different levels of sampling. While the accuracy fluctuates very slightly it essentially remains uniformly high for all levels of sampling. There are a number of possible reasons for the fact that the quality does not decrease:

sample size	time	quality
unlimited	6279 sec	.290
50000	1051 sec	.289
10000	336 sec	.291
5000	201 sec	.289
1000	60 sec	.288
500	35 sec	.288
100	10 sec	.288

Table 6: A comparison of retrieval times and translation quality when the number of translations is capped at various sample sizes

- The probability estimates under sampling are sufficiently good that the most probable translations remain unchanged,
- The interaction with the language model probability rules out the few misestimated probabilities, or
- The decoder tends to select longer or less frequent phrases which are not affected by the sampling.

While the translation quality remains essentially unchanged, the cumulative time that it takes to calculate the translation probabilities for all subphrases in the 430 sentence test set decreases radically. The total time drops by orders of magnitude from an hour and a half without sampling down to a mere 10 seconds with a cavalier amount of sampling. This suggests that the data structure is suitable for deployed SMT systems and that no additional caching need be done to compensate for the structure's computational complexity.

7 Discussion

The paper has presented a super-efficient data structure for phrase-based statistical machine translation. We have shown that current table-based methods are unwieldily when used in conjunction with large data sets and long phrases. We have contrasted this with our suffix array-based data structure which provides a very compact way of storing large data sets while simultaneously allowing the retrieval of arbitrarily long phrases.

For the NIST-2004 Arabic-English data set, which is among the largest currently assembled for statistical machine translation, our representation uses a very manageable 2 gigabytes of memory. This is less than is needed to store a table containing phrases with a maximum of three words, and is ten times less than the memory required to store a table with phrases of length eight.

We have further demonstrated that while computational complexity can make the retrieval of translation of frequent phrases slow, the use of sampling is an extremely effective countermeasure to this. We demonstrated that calculating phrase translation probabilities from sets of 100 occurrences or less results in nearly no decrease in translation quality.

The implications of the data structure presented in this paper are significant. The compact representation will allow us to easily scale to parallel corpora consisting of billions of words of text, and the retrieval of arbitrarily long phrases will allow experiments with alternative decoding strategies. These facts in combination allow for an even greater exploitation of training data in statistical machine translation.

References

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*.

Philipp Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation. Unpublished Draft.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*.

Udi Manber and Gene Myers. 1990. Suffix arrays: A new method for on-line string searches. In *The First Annual ACM-SIAM Symposium on Dicrete Algorithms*, pages 319–327.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–450, December.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Christoph Tillmann. 2003. A projection extension algorithm for statistical machine translation. In *Proceedings of EMNLP*.

Ashish Venugopal, Stephan Vogel, and Alex Waibel. 2003. Effective phrase translation extraction from alignment models. In *Proceedings of ACL*.

Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. 2003. The CMU statistical machine translation system. In *Proceedings of MT Summit 9*.

Mikio Yamamoto and Kenneth Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Compuatational Linguistics*, 27(1):1–30.

A Hierarchical Phrase-Based Model for Statistical Machine Translation

David Chiang
Institute for Advanced Computer Studies (UMIACS)
University of Maryland, College Park, MD 20742, USA
dchiang@umiacs.umd.edu

Abstract

We present a statistical phrase-based translation model that uses *hierarchical phrases*—phrases that contain subphrases. The model is formally a synchronous context-free grammar but is learned from a bitext without any syntactic information. Thus it can be seen as a shift to the *formal* machinery of syntax-based translation systems without any *linguistic* commitment. In our experiments using BLEU as a metric, the hierarchical phrase-based model achieves a relative improvement of 7.5% over Pharaoh, a state-of-the-art phrase-based system.

1 Introduction

The alignment template translation model (Och and Ney, 2004) and related phrase-based models advanced the previous state of the art by moving from words to *phrases* as the basic unit of translation. Phrases, which can be any substring and not necessarily phrases in any syntactic theory, allow these models to learn local reorderings, translation of short idioms, or insertions and deletions that are sensitive to local context. They are thus a simple and powerful mechanism for machine translation.

The basic phrase-based model is an instance of the noisy-channel approach (Brown et al., 1993),[1] in which the translation of a French sentence f into an English sentence e is modeled as:

$$(1) \quad \arg\max_e P(e \mid f) = \arg\max_e P(e, f)$$

$$(2) \quad = \arg\max_e (P(e) \times P(f \mid e))$$

The translation model $P(f \mid e)$ "encodes" e into f by the following steps:

1. segment e into phrases $\bar{e}_1 \cdots \bar{e}_I$, typically with a uniform distribution over segmentations;
2. reorder the $\bar{e}_i$ according to some distortion model;
3. translate each of the $\bar{e}_i$ into French phrases according to a model $P(\bar{f} \mid \bar{e})$ estimated from the training data.

Other phrase-based models model the joint distribution $P(e, f)$ (Marcu and Wong, 2002) or made $P(e)$ and $P(f \mid e)$ into features of a log-linear model (Och and Ney, 2002). But the basic architecture of phrase segmentation (or generation), phrase reordering, and phrase translation remains the same.

Phrase-based models can robustly perform translations that are localized to substrings that are common enough to have been observed in training. But Koehn et al. (2003) find that phrases longer than three words improve performance little, suggesting that data sparseness takes over for longer phrases. Above the phrase level, these models typically have a simple distortion model that reorders phrases independently of their content (Och and Ney, 2004; Koehn et al., 2003), or not at all (Zens and Ney, 2004; Kumar et al., 2005).

But it is often desirable to capture translations whose scope is larger than a few consecutive words.

[1] Throughout this paper, we follow the convention of Brown et al. of designating the source and target languages as "French" and "English," respectively. The variables f and e stand for source and target sentences; f_i^j stands for the substring of f from position i to position j inclusive, and similarly for e_i^j.

Consider the following Mandarin example and its English translation:

(3)

澳洲	是	与	北	韩	有	邦交
Aozhou	shi	yu	Bei	Han	you	bangjiao
Australia	is	with	North	Korea	have	dipl. rels.

的	少数	国家	之一
de	shaoshu	guojia	zhiyi
that	few	countries	one of

'Australia is one of the few countries that have diplomatic relations with North Korea'

If we count *zhiyi*, lit. 'of-one,' as a single token, then translating this sentence correctly into English requires reversing a sequence of five elements. When we run a phrase-based system, Pharaoh (Koehn et al., 2003; Koehn, 2004a), on this sentence (using the experimental setup described below), we get the following phrases with translations:

(4) [Aozhou] [shi] [yu] [Bei Han] [you] [bangjiao]$_1$ [de shaoshu guojia zhiyi]

[Australia] [is] [dipl. rels.]$_1$ [with] [North Korea] [is] [one of the few countries]

where we have used subscripts to indicate the reordering of phrases. The phrase-based model is able to order "diplomatic. . .Korea" correctly (using phrase reordering) and "one. . .countries" correctly (using a phrase translation), but does not accomplish the necessary inversion of those two groups. A lexicalized phrase-reordering model like that in use in ISI's system (Och et al., 2004) might be able to learn a better reordering, but simpler distortion models will probably not.

We propose a solution to these problems that does not interfere with the strengths of the phrase-based approach, but rather capitalizes on them: since phrases are good for learning reorderings of words, we can use them to learn reorderings of phrases as well. In order to do this we need *hierarchical phrases* that consist of both words and subphrases. For example, a hierarchical phrase pair that might help with the above example is:

(5) $\langle \text{yu } \boxed{1} \text{ you } \boxed{2}, \text{have } \boxed{2} \text{ with } \boxed{1} \rangle$

where $\boxed{1}$ and $\boxed{2}$ are placeholders for subphrases. This would capture the fact that Chinese PPs almost always modify VP on the left, whereas English PPs usually modify VP on the right. Because it generalizes over possible prepositional objects and direct objects, it acts both as a discontinuous phrase pair and as a phrase-reordering rule. Thus it is considerably more powerful than a conventional phrase pair.

Similarly,

(6) $\langle \boxed{1} \text{ de } \boxed{2}, \text{the } \boxed{2} \text{ that } \boxed{1} \rangle$

would capture the fact that Chinese relative clauses modify NPs on the left, whereas English relative clauses modify on the right; and

(7) $\langle \boxed{1} \text{ zhiyi}, \text{one of } \boxed{1} \rangle$

would render the construction *zhiyi* in English word order. These three rules, along with some conventional phrase pairs, suffice to translate the sentence correctly:

(8) [Aozhou] [shi] [[[yu [Bei Han]$_1$ you [bangjiao]$_2$] de [shaoshu guojia]$_3$] zhiyi]

[Australia] [is] [one of [the [few countries]$_3$ that [have [dipl. rels.]$_2$ with [North Korea]$_1$]]]

The system we describe below uses rules like this, and in fact is able to learn them automatically from a bitext without syntactic annotation. It translates the above example almost exactly as we have shown, the only error being that it omits the word 'that' from (6) and therefore (8).

These hierarchical phrase pairs are formally productions of a synchronous context-free grammar (defined below). A move to synchronous CFG can be seen as a move towards syntax-based MT; however, we make a distinction here between *formally* syntax-based and *linguistically* syntax-based MT. A system like that of Yamada and Knight (2001) is both formally and linguistically syntax-based: formally because it uses synchronous CFG, linguistically because the structures it is defined over are (on the English side) informed by syntactic theory (via the Penn Treebank). Our system is formally syntax-based in that it uses synchronous CFG, but not necessarily linguistically syntax-based, because it induces a grammar from a parallel text without relying on any linguistic annotations or assumptions; the result sometimes resembles a syntactician's grammar but often does not. In this respect it resembles Wu's bilingual bracketer (Wu, 1997), but ours uses a different extraction method that allows more than one lexical item in a rule, in keeping with the phrase-based philosophy. Our extraction method is basically the same as that of Block (2000), except we allow more than one nonterminal symbol in a rule, and use a more sophisticated probability model.

In this paper we describe the design and implementation of our hierarchical phrase-based model, and report on experiments that demonstrate that hierarchical phrases indeed improve translation.

2 The model

Our model is based on a weighted synchronous CFG (Aho and Ullman, 1969). In a synchronous CFG the elementary structures are rewrite rules with aligned pairs of right-hand sides:

(9) $$X \rightarrow \langle \gamma, \alpha, \sim \rangle$$

where X is a nonterminal, γ and α are both strings of terminals and nonterminals, and $\sim$ is a one-to-one correspondence between nonterminal occurrences in γ and nonterminal occurrences in α. Rewriting begins with a pair of linked start symbols. At each step, two coindexed nonterminals are rewritten using the two components of a single rule, such that none of the newly introduced symbols is linked to any symbols already present.

Thus the hierarchical phrase pairs from our above example could be formalized in a synchronous CFG as:

(10) $$\mathrm{X} \rightarrow \langle \text{yu } \mathrm{X}_{\boxed{1}} \text{ you } \mathrm{X}_{\boxed{2}}, \text{have } \mathrm{X}_{\boxed{2}} \text{ with } \mathrm{X}_{\boxed{1}} \rangle$$

(11) $$\mathrm{X} \rightarrow \langle \mathrm{X}_{\boxed{1}} \text{ de } \mathrm{X}_{\boxed{2}}, \text{the } \mathrm{X}_{\boxed{2}} \text{ that } \mathrm{X}_{\boxed{1}} \rangle$$

(12) $$\mathrm{X} \rightarrow \langle \mathrm{X}_{\boxed{1}} \text{ zhiyi}, \text{one of } \mathrm{X}_{\boxed{1}} \rangle$$

where we have used boxed indices to indicate which occurrences of X are linked by $\sim$.

Note that we have used only a single nonterminal symbol X instead of assigning syntactic categories to phrases. In the grammar we extract from a bitext (described below), all of our rules use only X, except for two special "glue" rules, which combine a sequence of Xs to form an S:

(13) $$\mathrm{S} \rightarrow \langle \mathrm{S}_{\boxed{1}} \mathrm{X}_{\boxed{2}}, \mathrm{S}_{\boxed{1}} \mathrm{X}_{\boxed{2}} \rangle$$

(14) $$\mathrm{S} \rightarrow \langle \mathrm{X}_{\boxed{1}}, \mathrm{X}_{\boxed{1}} \rangle$$

These give the model the option to build only partial translations using hierarchical phrases, and then combine them serially as in a standard phrase-based model. For a partial example of a synchronous CFG derivation, see Figure 1.

Following Och and Ney (2002), we depart from the traditional noisy-channel approach and use a more general log-linear model. The weight of each rule is:

(15) $$w(X \rightarrow \langle \gamma, \alpha \rangle) = \prod_i \phi_i(X \rightarrow \langle \gamma, \alpha \rangle)^{\lambda_i}$$

where the ϕ_i are features defined on rules. For our experiments we used the following features, analogous to Pharaoh's default feature set:

- $P(\gamma \mid \alpha)$ and $P(\alpha \mid \gamma)$, the latter of which is not found in the noisy-channel model, but has been previously found to be a helpful feature (Och and Ney, 2002);
- the lexical weights $P_w(\gamma \mid \alpha)$ and $P_w(\alpha \mid \gamma)$ (Koehn et al., 2003), which estimate how well the words in α translate the words in γ;[2]
- a phrase penalty exp(1), which allows the model to learn a preference for longer or shorter derivations, analogous to Koehn's phrase penalty (Koehn, 2003).

The exceptions to the above are the two glue rules, (13), which has weight one, and (14), which has weight

(16) $$w(\mathrm{S} \rightarrow \langle \mathrm{S}_{\boxed{1}} \mathrm{X}_{\boxed{2}}, \mathrm{S}_{\boxed{1}} \mathrm{X}_{\boxed{2}} \rangle) = \exp(-\lambda_g)$$

the idea being that λ_g controls the model's preference for hierarchical phrases over serial combination of phrases.

Let D be a derivation of the grammar, and let $f(D)$ and $e(D)$ be the French and English strings generated by D. Let us represent D as a set of triples $\langle r, i, j \rangle$, each of which stands for an application of a grammar rule r to rewrite a nonterminal that spans $f(D)_i^j$ on the French side.[3] Then the weight of D

[2]This feature uses word alignment information, which is discarded in the final grammar. If a rule occurs in training with more than one possible word alignment, Koehn et al. take the maximum lexical weight; we take a weighted average.

[3]This representation is not completely unambiguous, but is sufficient for defining the model.

$$
\begin{aligned}
\langle \mathrm{S}_{\boxed{1}}, \mathrm{S}_{\boxed{1}} \rangle &\Rightarrow \langle \mathrm{S}_{\boxed{2}} \mathrm{X}_{\boxed{3}}, \mathrm{S}_{\boxed{2}} \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \mathrm{S}_{\boxed{4}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}}, \mathrm{S}_{\boxed{4}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \mathrm{X}_{\boxed{6}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}}, \mathrm{X}_{\boxed{6}} \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \text{Aozhou } \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}}, \text{Australia } \mathrm{X}_{\boxed{5}} \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \text{Aozhou shi } \mathrm{X}_{\boxed{3}}, \text{Australia is } \mathrm{X}_{\boxed{3}} \rangle \\
&\Rightarrow \langle \text{Aozhou shi } \mathrm{X}_{\boxed{7}} \text{ zhiyi}, \text{Australia is one of } \mathrm{X}_{\boxed{7}} \rangle \\
&\Rightarrow \langle \text{Aozhou shi } \mathrm{X}_{\boxed{8}} \text{ de } \mathrm{X}_{\boxed{9}} \text{ zhiyi}, \text{Australia is one of the } \mathrm{X}_{\boxed{9}} \text{ that } \mathrm{X}_{\boxed{8}} \rangle \\
&\Rightarrow \langle \text{Aozhou shi yu } \mathrm{X}_{\boxed{1}} \text{ you } \mathrm{X}_{\boxed{2}} \text{ de } \mathrm{X}_{\boxed{9}} \text{ zhiyi}, \text{Australia is one of the } \mathrm{X}_{\boxed{9}} \text{ that have } \mathrm{X}_{\boxed{2}} \text{ with } \mathrm{X}_{\boxed{1}} \rangle
\end{aligned}
$$

Figure 1: Example partial derivation of a synchronous CFG.

is the product of the weights of the rules used in the translation, multiplied by the following extra factors:

$$(17)\quad w(D) = \prod_{\langle r,i,j\rangle \in D} w(r) \times p_{lm}(e)^{\lambda_{lm}} \times \exp(-\lambda_{wp}|e|)$$

where p_{lm} is the language model, and $\exp(-\lambda_{wp}|e|)$, the word penalty, gives some control over the length of the English output.

We have separated these factors out from the rule weights for notational convenience, but it is conceptually cleaner (and necessary for polynomial-time decoding) to integrate them into the rule weights, so that the whole model is a weighted synchronous CFG. The word penalty is easy; the language model is integrated by intersecting the English-side CFG with the language model, which is a weighted finite-state automaton.

3 Training

The training process begins with a word-aligned corpus: a set of triples $\langle f, e, \sim\rangle$, where f is a French sentence, e is an English sentence, and $\sim$ is a (many-to-many) binary relation between positions of f and positions of e. We obtain the word alignments using the method of Koehn et al. (2003), which is based on that of Och and Ney (2004). This involves running GIZA++ (Och and Ney, 2000) on the corpus in both directions, and applying refinement rules (the variant they designate "final-and") to obtain a single many-to-many word alignment for each sentence.

Then, following Och and others, we use heuristics to hypothesize a distribution of possible derivations of each training example, and then estimate the phrase translation parameters from the hypothesized distribution. To do this, we first identify *initial phrase* pairs using the same criterion as previous systems (Och and Ney, 2004; Koehn et al., 2003):

Definition 1. Given a word-aligned sentence pair $\langle f, e, \sim\rangle$, a rule $\langle f_i^j, e_{i'}^{j'}\rangle$ is an initial phrase pair of $\langle f, e, \sim\rangle$ iff:

1. $f_k \sim e_{k'}$ for some $k \in [i, j]$ and $k' \in [i', j']$;
2. $f_k \nsim e_{k'}$ for all $k \in [i, j]$ and $k' \notin [i', j']$;
3. $f_k \nsim e_{k'}$ for all $k \notin [i, j]$ and $k' \in [i', j']$.

Next, we form all possible differences of phrase pairs:

Definition 2. The set of rules of $\langle f, e, \sim\rangle$ is the smallest set satisfying the following:

1. If $\langle f_i^j, e_{i'}^{j'}\rangle$ is an initial phrase pair, then
$$\mathrm{X} \to \langle f_i^j, e_{i'}^{j'}\rangle$$
is a rule.
2. If $r = \mathrm{X} \to \langle \gamma, \alpha\rangle$ is a rule and $\langle f_i^j, e_{i'}^{j'}\rangle$ is an initial phrase pair such that $\gamma = \gamma_1 f_i^j \gamma_2$ and $\alpha = \alpha_1 e_{i'}^{j'} \alpha_2$, then
$$\mathrm{X} \to \langle \gamma_1 \mathrm{X}_{\boxed{k}} \gamma_2, \alpha_1 \mathrm{X}_{\boxed{k}} \alpha_2\rangle$$
is a rule, where k is an index not used in r.

The above scheme generates a very large number of rules, which is undesirable not only because it makes training and decoding very slow, but also because it creates *spurious ambiguity*—a situation where the decoder produces many derivations that are distinct yet have the same model feature vectors and give the same translation. This can result in *n*-best lists with very few different translations or feature vectors, which is problematic for the algorithm we use to tune the feature weights. Therefore we filter our grammar according to the following principles, chosen to balance grammar size and performance on our development set:

1. If there are multiple initial phrase pairs containing the same set of alignment points, we keep only the smallest.

2. Initial phrases are limited to a length of 10 on the French side, and rule to five (nonterminals plus terminals) on the French right-hand side.

3. In the subtraction step, f_i^j must have length greater than one. The rationale is that little would be gained by creating a new rule that is no shorter than the original.

4. Rules can have at most two nonterminals, which simplifies the decoder implementation. Moreover, we prohibit nonterminals that are adjacent on the French side, a major cause of spurious ambiguity.

5. A rule must have at least one pair of aligned words, making translation decisions always based on some lexical evidence.

Now we must hypothesize weights for all the derivations. Och's method gives equal weight to all the extracted phrase occurences. However, our method may extract many rules from a single initial phrase pair; therefore we distribute weight equally among initial phrase pairs, but distribute that weight equally among the rules extracted from each. Treating this distribution as our observed data, we use relative-frequency estimation to obtain $P(\gamma \mid \alpha)$ and $P(\alpha \mid \gamma)$.

4 Decoding

Our decoder is a CKY parser with beam search together with a postprocessor for mapping French derivations to English derivations. Given a French sentence f, it finds the best derivation (or n best derivations, with little overhead) that generates $\langle f, e \rangle$ for some e. Note that we find the English yield of the highest-probability single derivation

$$e\left(\underset{D \text{ s.t. } f(D) = f}{\arg\max}\; w(D)\right) \tag{18}$$

and not necessarily the highest-probability e, which would require a more expensive summation over derivations.

We prune the search space in several ways. First, an item that has a score worse than β times the best score in the same cell is discarded; second, an item that is worse than the bth best item in the same cell is discarded. Each cell contains all the items standing for X spanning f_i^j. We choose b and β to balance speed and performance on our development set. For our experiments, we set $b = 40, \beta = 10^{-1}$ for X cells, and $b = 15, \beta = 10^{-1}$ for S cells. We also prune rules that have the same French side ($b = 100$).

The parser only operates on the French-side grammar; the English-side grammar affects parsing only by increasing the effective grammar size, because there may be multiple rules with the same French side but different English sides, and also because intersecting the language model with the English-side grammar introduces many states into the nonterminal alphabet, which are projected over to the French side. Thus, our decoder's search space is many times larger than a monolingual parser's would be. To reduce this effect, we apply the following heuristic when filling a cell: if an item falls outside the beam, then any item that would be generated using a lower-scoring rule or a lower-scoring antecedent item is also assumed to fall outside the beam. This heuristic greatly increases decoding speed, at the cost of some search errors.

Finally, the decoder has a constraint that prohibits any X from spanning a substring longer than 10 on the French side, corresponding to the maximum length constraint on initial rules during training. This makes the decoding algorithm asymptotically linear-time.

The decoder is implemented in Python, an interpreted language, with C++ code from the SRI Language Modeling Toolkit (Stolcke, 2002). Using the settings described above, on a 2.4 GHz Pentium IV, it takes about 20 seconds to translate each sentence (average length about 30). This is faster than our

Python implementation of a standard phrase-based decoder, so we expect that a future optimized implementation of the hierarchical decoder will run at a speed competitive with other phrase-based systems.

5 Experiments

Our experiments were on Mandarin-to-English translation. We compared a baseline system, the state-of-the-art phrase-based system Pharaoh (Koehn et al., 2003; Koehn, 2004a), against our system. For all three systems we trained the translation model on the FBIS corpus (7.2M+9.2M words); for the language model, we used the SRI Language Modeling Toolkit to train a trigram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998) on 155M words of English newswire text, mostly from the Xinhua portion of the Gigaword corpus. We used the 2002 NIST MT evaluation test set as our development set, and the 2003 test set as our test set. Our evaluation metric was BLEU (Papineni et al., 2002), as calculated by the NIST script (version 11a) with its default settings, which is to perform case-insensitive matching of *n*-grams up to $n = 4$, and to use the shortest (as opposed to nearest) reference sentence for the brevity penalty. The results of the experiments are summarized in Table 1.

5.1 Baseline

The baseline system we used for comparison was Pharaoh (Koehn et al., 2003; Koehn, 2004a), as publicly distributed. We used the default feature set: language model (same as above), $p(\overline{f} \mid \overline{e})$, $p(\overline{e} \mid \overline{f})$, lexical weighting (both directions), distortion model, word penalty, and phrase penalty. We ran the trainer with its default settings (maximum phrase length 7), and then used Koehn's implementation of minimum-error-rate training (Och, 2003) to tune the feature weights to maximize the system's BLEU score on our development set, yielding the values shown in Table 2. Finally, we ran the decoder on the test set, pruning the phrase table with $b = 100$, pruning the chart with $b = 100, \beta = 10^{-5}$, and limiting distortions to 4. These are the default settings, except for the phrase table's *b*, which was raised from 20, and the distortion limit. Both of these changes, made by Koehn's minimum-error-rate trainer by default, improve performance on the development set.

Rank	**Chinese**	**English**
1	。	.
3	的	the
14	在	in
23	的	's
577	$X_{\boxed{1}}$ 的 $X_{\boxed{2}}$	the $X_{\boxed{2}}$ of $X_{\boxed{1}}$
735	$X_{\boxed{1}}$ 的 $X_{\boxed{2}}$	the $X_{\boxed{2}}$ $X_{\boxed{1}}$
763	$X_{\boxed{1}}$ 之一	one of $X_{\boxed{1}}$
1201	$X_{\boxed{1}}$ 总统	president $X_{\boxed{1}}$
1240	$X_{\boxed{1}}$ 美元	$\$ X_{\boxed{1}}$
2091	今年 $X_{\boxed{1}}$	$X_{\boxed{1}}$ this year
3253	百分之 $X_{\boxed{1}}$	$X_{\boxed{1}}$ percent
10508	在 $X_{\boxed{1}}$ 下	under $X_{\boxed{1}}$
28426	在 $X_{\boxed{1}}$ 前	before $X_{\boxed{1}}$
47015	$X_{\boxed{1}}$ 的 $X_{\boxed{2}}$	the $X_{\boxed{2}}$ that $X_{\boxed{1}}$
1752457	与 $X_{\boxed{1}}$ 有 $X_{\boxed{2}}$	have $X_{\boxed{2}}$ with $X_{\boxed{1}}$

Figure 2: A selection of extracted rules, with ranks after filtering for the development set. All have X for their left-hand sides.

5.2 Hierarchical model

We ran the training process of Section 3 on the same data, obtaining a grammar of 24M rules. When filtered for the development set, the grammar has 2.2M rules (see Figure 2 for examples). We then ran the minimum-error rate trainer with our decoder to tune the feature weights, yielding the values shown in Table 2. Note that λ_g penalizes the glue rule much less than λ_{pp} does ordinary rules. This suggests that the model will prefer serial combination of phrases, unless some other factor supports the use of hierarchical phrases (e.g., a better language model score).

We then tested our system, using the settings described above.[4] Our system achieves an absolute improvement of 0.02 over the baseline (7.5% relative), without using any additional training data. This difference is statistically significant ($p < 0.01$).[5] See Table 1, which also shows that the relative gain is higher for higher *n*-grams.

[4]Note that we gave Pharaoh wider beam settings than we used on our own decoder; on the other hand, since our decoder's chart has more cells, its *b* limits do not need to be as high.

[5]We used Zhang's significance tester (Zhang et al., 2004), which uses bootstrap resampling (Koehn, 2004b); it was modified to conform to NIST's current definition of the BLEU brevity penalty.

	BLEU-*n*	***n*-gram precisions**							
System	4	1	2	3	4	5	6	7	8
Pharaoh	0.2676	0.72	0.37	0.19	0.10	0.052	0.027	0.014	0.0075
hierarchical	0.2877	0.74	0.39	0.21	0.11	0.060	0.032	0.017	0.0084
+constituent	0.2881	0.73	0.39	0.21	0.11	0.062	0.032	0.017	0.0088

Table 1: Results on baseline system and hierarchical system, with and without constituent feature.

	Features									
System	$P_{lm}(e)$	$P(\gamma\|\alpha)$	$P(\alpha\|\gamma)$	$P_w(\gamma\|\alpha)$	$P_w(\alpha\|\gamma)$	Word	Phr	λ_d	λ_g	λ_c
Pharaoh	0.19	0.095	0.030	0.14	0.029	−0.20	0.22	0.11	—	—
hierarchical	0.15	0.036	0.074	0.037	0.076	−0.32	0.22	—	0.09	—
+constituent	0.11	0.026	0.062	0.025	0.029	−0.23	0.21	—	0.11	0.20

Table 2: Feature weights obtained by minimum-error-rate training (normalized so that absolute values sum to one). Word = word penalty; Phr = phrase penalty. Note that we have inverted the sense of Pharaoh's phrase penalty so that a positive weight indicates a penalty.

5.3 Adding a constituent feature

The use of hierarchical structures opens the possibility of making the model sensitive to syntactic structure. Koehn et al. (2003) mention German ⟨es gibt, there is⟩ as an example of a good phrase pair which is not a syntactic phrase pair, and report that favoring syntactic phrases does not improve accuracy. But in our model, the rule

(19) $$X \rightarrow \langle \text{es gibt } X_{\boxed{1}}, \text{there is } X_{\boxed{1}} \rangle$$

would indeed respect syntactic phrases, because it builds a pair of Ss out of a pair of NPs. Thus, favoring subtrees in our model that are syntactic phrases might provide a fairer way of testing the hypothesis that syntactic phrases are better phrases.

This feature adds a factor to (17),

(20) $$c(i,j) = \begin{cases} 1 & \text{if } f_i^j \text{ is a constituent} \\ 0 & \text{otherwise} \end{cases}$$

as determined by a statistical tree-substitution-grammar parser (Bikel and Chiang, 2000), trained on the Penn Chinese Treebank, version 3 (250k words). Note that the parser was run only on the test data and not the (much larger) training data. Rerunning the minimum-error-rate trainer with the new feature yielded the feature weights shown in Table 2. Although the feature improved accuracy on the development set (from 0.314 to 0.322), it gave no statistically significant improvement on the test set.

6 Conclusion

Hierarchical phrase pairs, which can be learned without any syntactically-annotated training data, improve translation accuracy significantly compared with a state-of-the-art phrase-based system. They also facilitate the incorporation of syntactic information, which, however, did not provide a statistically significant gain.

Our primary goal for the future is to move towards a more syntactically-motivated grammar, whether by automatic methods to induce syntactic categories, or by better integration of parsers trained on annotated data. This would potentially improve both accuracy and efficiency. Moreover, reducing the grammar size would allow more ambitious training settings. The maximum initial phrase length is currently 10; preliminary experiments show that increasing this limit to as high as 15 does improve accuracy, but requires more memory. On the other hand, we have successfully trained on almost 30M+30M words by tightening the initial phrase length limit for part of the data. Streamlining the grammar would allow further experimentation in these directions.

In any case, future improvements to this system will maintain the design philosophy proven here, that ideas from syntax should be incorporated into statistical translation, but not in exchange for the strengths of the phrase-based approach.

Acknowledgements

I would like to thank Philipp Koehn for the use of the Pharaoh software; and Adam Lopez, Michael Subotin, Nitin Madnani, Christof Monz, Liang Huang, and Philip Resnik. This work was partially supported by ONR MURI contract FCPO.810548265 and Department of Defense contract RD-02-5700. *S. D. G.*

References

A. V. Aho and J. D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3:37–56.

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the Chinese Treebank. In *Proceedings of the Second Chinese Language Processing Workshop*, pages 1–6.

Hans Ulrich Block. 2000. Example-based incremental synchronous interpretation. In Wolfgang Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 411–417. Springer-Verlag, Berlin.

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*, 19:263–311.

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.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133.

Philipp Koehn. 2003. *Noun Phrase Translation*. Ph.D. thesis, University of Southern California.

Philipp Koehn. 2004a. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the Sixth Conference of the Association for Machine Translation in the Americas*, pages 115–124.

Philipp Koehn. 2004b. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395.

Shankar Kumar, Yonggang Deng, and William Byrne. 2005. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*. To appear.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–139.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the ACL*, pages 440–447.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 295–302.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.

Franz Josef Och, Ignacio Thayer, Daniel Marcu, Kevin Knight, Dragos Stefan Munteanu, Quamrul Tipu, Michel Galley, and Mark Hopkins. 2004. Arabic and Chinese MT at USC/ISI. Presentation given at NIST Machine Translation Evaluation Workshop.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 160–167.

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 ACL*, pages 311–318.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the ACL*, pages 523–530.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of HLT-NAACL 2004*, pages 257–264.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC)*, pages 2051–2054.

Dependency Treelet Translation: Syntactically Informed Phrasal SMT

Chris Quirk, Arul Menezes
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{chrisq,arulm}@microsoft.com

Colin Cherry
University of Alberta
Edmonton, Alberta
Canada T6G 2E1
colinc@cs.ualberta.ca

Abstract

We describe a novel approach to statistical machine translation that combines syntactic information in the source language with recent advances in phrasal translation. This method requires a source-language dependency parser, target language word segmentation and an unsupervised word alignment component. We align a parallel corpus, project the source dependency parse onto the target sentence, extract dependency treelet translation pairs, and train a tree-based ordering model. We describe an efficient decoder and show that using these tree-based models in combination with conventional SMT models provides a promising approach that incorporates the power of phrasal SMT with the linguistic generality available in a parser.

1. Introduction

Over the past decade, we have witnessed a revolution in the field of machine translation (MT) toward statistical or corpus-based methods. Yet despite this success, statistical machine translation (SMT) has many hurdles to overcome. While it excels at translating domain-specific terminology and fixed phrases, grammatical generalizations are poorly captured and often mangled during translation (Thurmair, 04).

1.1. Limitations of string-based phrasal SMT

State-of-the-art phrasal SMT systems such as (Koehn et al., 03) and (Vogel et al., 03) model translations of *phrases* (here, strings of adjacent words, not syntactic constituents) rather than individual words. Arbitrary reordering of words is allowed within memorized phrases, but typically only a small amount of phrase reordering is allowed, modeled in terms of offset positions at the string level. This reordering model is very limited in terms of linguistic generalizations. For instance, when translating English to Japanese, an ideal system would automatically learn large-scale typological differences: English SVO clauses generally become Japanese SOV clauses, English post-modifying prepositional phrases become Japanese pre-modifying postpositional phrases, etc. A phrasal SMT system may learn the internal reordering of specific common phrases, but it cannot generalize to unseen phrases that share the same linguistic structure.

In addition, these systems are limited to phrases contiguous in both source and target, and thus cannot learn the generalization that English *not* may translate as French *ne…pas* except in the context of specific intervening words.

1.2. Previous work on syntactic SMT[1]

The hope in the SMT community has been that the incorporation of syntax would address these issues, but that promise has yet to be realized.

One simple means of incorporating syntax into SMT is by re-ranking the *n*-best list of a baseline SMT system using various syntactic models, but Och et al. (04) found very little positive impact with this approach. However, an *n*-best list of even 16,000 translations captures only a tiny fraction of the ordering possibilities of a 20 word sentence; re-ranking provides the syntactic model no opportunity to boost or prune large sections of that search space.

Inversion Transduction Grammars (Wu, 97), or ITGs, treat translation as a process of parallel parsing of the source and target language via a synchronized grammar. To make this process

[1] Note that since this paper does not address the word alignment problem directly, we do not discuss the large body of work on incorporating syntactic information into the word alignment process.

computationally efficient, however, some severe simplifying assumptions are made, such as using a single non-terminal label. This results in the model simply learning a very high level preference regarding how often nodes should switch order without any contextual information. Also these translation models are intrinsically word-based; phrasal combinations are not modeled directly, and results have not been competitive with the top phrasal SMT systems.

Along similar lines, Alshawi et al. (2000) treat translation as a process of simultaneous induction of source and target dependency trees using head-transduction; again, no separate parser is used.

Yamada and Knight (01) employ a parser in the target language to train probabilities on a set of operations that convert a target language tree to a source language string. This improves fluency slightly (Charniak et al., 03), but fails to significantly impact overall translation quality. This may be because the parser is applied to MT output, which is notoriously unlike native language, and no additional insight is gained via source language analysis.

Lin (04) translates dependency trees using paths. This is the first attempt to incorporate large phrasal SMT-style memorized patterns together with a separate source dependency parser and SMT models. However the phrases are limited to linear paths in the tree, the only SMT model used is a maximum likelihood channel model and there is no ordering model. Reported BLEU scores are far below the leading phrasal SMT systems.

MSR-MT (Menezes & Richardson, 01) parses both source and target languages to obtain a logical form (LF), and translates source LFs using memorized aligned LF patterns to produce a target LF. It utilizes a separate sentence realization component (Ringger et al., 04) to turn this into a target sentence. As such, it does not use a target language model during decoding, relying instead on MLE channel probabilities and heuristics such as pattern size. Recently Aue et al. (04) incorporated an LF-based language model (LM) into the system for a small quality boost. A key disadvantage of this approach and related work (Ding & Palmer, 02) is that it requires a parser in *both* languages, which severely limits the language pairs that can be addressed.

2. Dependency Treelet Translation

In this paper we propose a novel dependency tree-based approach to phrasal SMT which uses tree-based 'phrases' and a tree-based ordering model in combination with conventional SMT models to produce state-of-the-art translations.

Our system employs a source-language dependency parser, a target language word segmentation component, and an unsupervised word alignment component to learn treelet translations from a parallel sentence-aligned corpus. We begin by parsing the source text to obtain dependency trees and word-segmenting the target side, then applying an off-the-shelf word alignment component to the bitext.

The word alignments are used to project the source dependency parses onto the target sentences. From this aligned parallel dependency corpus we extract a treelet translation model incorporating source and target treelet pairs, where a *treelet* is defined to be an arbitrary connected subgraph of the dependency tree. A unique feature is that we allow treelets with a wildcard root, effectively allowing mappings for siblings in the dependency tree. This allows us to model important phenomena, such as *not …* → *ne…pas*. We also train a variety of statistical models on this aligned dependency tree corpus, including a channel model and an order model.

To translate an input sentence, we parse the sentence, producing a dependency tree for that sentence. We then employ a decoder to find a combination and ordering of treelet translation pairs that cover the source tree and are optimal according to a set of models that are combined in a log-linear framework as in (Och, 03).

This approach offers the following advantages over string-based SMT systems: Instead of limiting learned phrases to contiguous word sequences, we allow translation by all possible phrases that form connected subgraphs (treelets) in the source and target dependency trees. This is a powerful extension: the vast majority of surface-contiguous phrases are also treelets of the tree; in addition, we gain discontiguous phrases, including combinations such as verb-object, article-noun, adjective-noun etc. regardless of the number of intervening words.

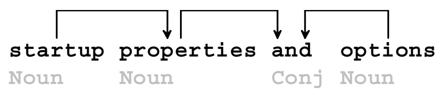


Figure 1. An example dependency tree.

Another major advantage is the ability to employ more powerful models for reordering source language constituents. These models can incorporate information from the source analysis. For example, we may model directly the probability that the translation of an object of a preposition in English should precede the corresponding postposition in Japanese, or the probability that a pre-modifying adjective in English translates into a post-modifier in French.

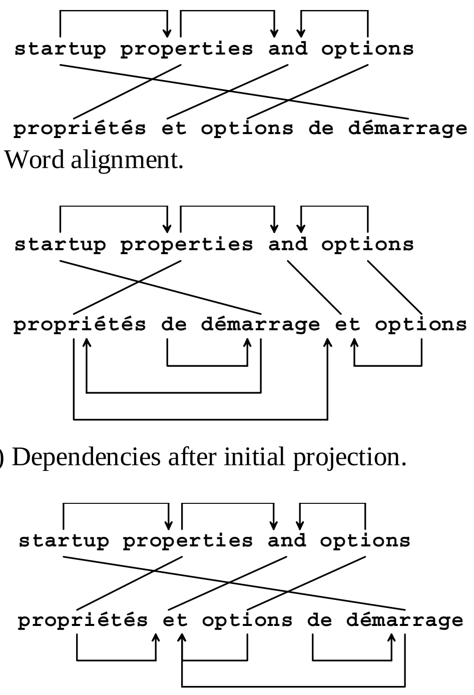
2.1. Parsing and alignment

We require a source language dependency parser that produces unlabeled, ordered dependency trees and annotates each source word with a part-of-speech (POS). An example dependency tree is shown in Figure 1. The arrows indicate the head annotation, and the POS for each candidate is listed underneath. For the target language we only require word segmentation.

To obtain word alignments we currently use GIZA++ (Och & Ney, 03). We follow the common practice of deriving many-to-many alignments by running the IBM models in both directions and combining the results heuristically. Our heuristics differ in that they constrain many-to-one alignments to be contiguous in the source dependency tree. A detailed description of these heuristics can be found in Quirk et al. (04).

2.2. Projecting dependency trees

Given a word aligned sentence pair and a source dependency tree, we use the alignment to project the source structure onto the target sentence. One-to-one alignments project directly to create a target tree isomorphic to the source. Many-to-one alignments project similarly; since the 'many' source nodes are connected in the tree, they act as if condensed into a single node. In the case of one-to-many alignments we project the source node to the rightmost[2] of the 'many' target words, and make the rest of the target words dependent on it.


startup properties and options
propriétés et options de démarrage
startup properties and options
propriétés de démarrage et options
startup properties and options
propriétés et options de démarrage

(a) Word alignment.

(b) Dependencies after initial projection.

(c) Dependencies after reattachment step.

Figure 2. Projection of dependencies.

Unaligned target words[3] are attached into the dependency structure as follows: assume there is an unaligned word t_j in position j. Let $i < j$ and $k > j$ be the target positions closest to j such that t_i depends on t_k or vice versa: attach t_j to the lower of t_i or t_k. If all the nodes to the left (or right) of position j are unaligned, attach t_j to the left-most (or right-most) word that is aligned.

The target dependency tree created in this process may not read off in the same order as the target string, since our alignments do not enforce phrasal cohesion. For instance, consider the projection of the parse in Figure 1 using the word alignment in Figure 2a. Our algorithm produces the dependency tree in Figure 2b. If we read off the leaves in a left-to-right in-order traversal, we do not get the original input string: *de démarrage* appears in the wrong place.

A second reattachment pass corrects this situation. For each node in the wrong order, we reattach it to the lowest of its ancestors such that it is in the correct place relative to its siblings and parent. In Figure 2c, reattaching *démarrage* to *et* suffices to produce the correct order.

[2] If the target language is Japanese, leftmost may be more appropriate.

[3] Source unaligned nodes do not present a problem, with the exception that if the root is unaligned, the projection process produces a forest of target trees anchored by a dummy root.

2.3. Extracting treelet translation pairs

From the aligned pairs of dependency trees we extract all pairs of aligned source and target treelets along with word-level alignment linkages, up to a configurable maximum size. We also keep treelet counts for maximum likelihood estimation.

2.4. Order model

Phrasal SMT systems often use a model to score the ordering of a set of phrases. One approach is to penalize any deviation from monotone decoding; another is to estimate the probability that a source phrase in position *i* translates to a target phrase in position *j* (Koehn et al., 03).

We attempt to improve on these approaches by incorporating syntactic information. Our model assigns a probability to the order of a target tree given a source tree. Under the assumption that constituents generally move as a whole, we predict the probability of each given ordering of modifiers independently. That is, we make the following simplifying assumption (where *c* is a function returning the set of nodes modifying *t*):

$$\mathrm{P}(order(T) \mid S,T) = \prod_{t \in T} \mathrm{P}(order(c(t)) \mid S,T)$$

Furthermore, we assume that the position of each child can be modeled independently in terms of a head-relative position:

$$\mathrm{P}(order(c(t)) \mid S,T) = \prod_{m \in c(t)} \mathrm{P}(pos(m,t) \mid S,T)$$

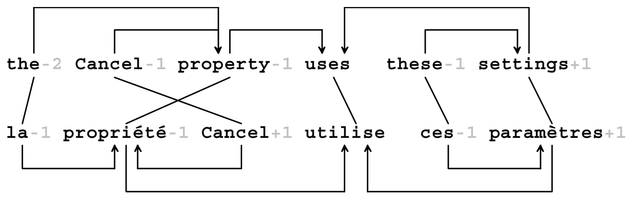
Figure 3a demonstrates an aligned dependency tree pair annotated with head-relative positions; Figure 3b presents the same information in an alternate tree-like representation.

We currently use a small set of features reflecting very local information in the dependency tree to model P(pos(*m*,*t*) | *S*, *T*):

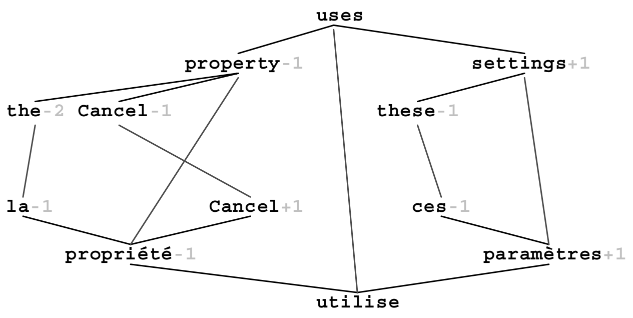
- The lexical items of the head and modifier.
- The lexical items of the source nodes aligned to the head and modifier.
- The part-of-speech ("cat") of the source nodes aligned to the head and modifier.
- The head-relative position of the source node aligned to the source modifier. [4]

As an example, consider the children of *propriété* in Figure 3. The head-relative positions

[4] One can also include features of siblings to produce a Markov ordering model. However, we found that this had little impact in practice.



(a) Head annotation representation



(b) Branching structure representation.

Figure 3. Aligned dependency tree pair, annotated with head-relative positions

of its modifiers *la* and *Cancel* are -1 and +1, respectively. Thus we try to predict as follows:

P(pos(m_1) = -1 |
lex(m_1)="*la*", lex(*h*)="*propriété*",
lex(src(m_1))="*the*", lex(src(*h*)="*property*",
cat(src(m_1))=Determiner, cat(src(*h*))=Noun,
position(src(m_1))=-2) ·
P(pos(m_2) = +1 |
lex(m_2)="*Cancel*", lex(*h*)="*propriété*",
lex(src(m_2))="*Cancel*", lex(src(h))="*property*",
cat(src(m_2))=Noun, cat(src(h))=Noun,
position(src(m_2))=-1)

The training corpus acts as a supervised training set: we extract a training feature vector from each of the target language nodes in the aligned dependency tree pairs. Together these feature vectors are used to train a decision tree (Chickering, 02). The distribution at each leaf of the DT can be used to assign a probability to each possible target language position. A more detailed description is available in (Quirk et al., 04).

2.5. Other models

Channel Models: We incorporate two distinct channel models, a maximum likelihood estimate (MLE) model and a model computed using Model-1 word-to-word alignment probabilities as in (Vogel et al., 03). The MLE model effectively captures non-literal phrasal translations such as idioms, but suffers from data sparsity. The word-to-word model does not typically suffer from data sparsity, but prefers more literal translations.

Given a set of treelet translation pairs that cover a given input dependency tree and produce a target dependency tree, we model the probability of source given target as the product of the individual treelet translation probabilities: we assume a uniform probability distribution over the decompositions of a tree into treelets.

Target Model: Given an ordered target language dependency tree, it is trivial to read off the surface string. We evaluate this string using a trigram model with modified Kneser-Ney smoothing.

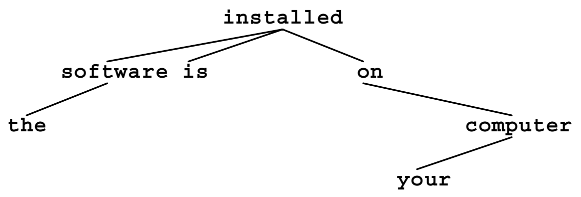
Miscellaneous Feature Functions: The log-linear framework allows us to incorporate other feature functions as 'models' in the translation process. For instance, using fewer, larger treelet translation pairs often provides better translations, since they capture more context and allow fewer possibilities for search and model error. Therefore we add a feature function that counts the number of phrases used. We also add a feature that counts the number of target words; this acts as an insertion/deletion bonus/penalty.

3. Decoding

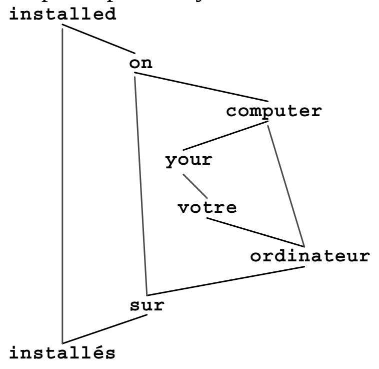
The challenge of tree-based decoding is that the traditional left-to-right decoding approach of string-based systems is inapplicable. Additional challenges are posed by the need to handle treelets—perhaps discontiguous or overlapping—and a combinatorially explosive ordering space.

Our decoding approach is influenced by ITG (Wu, 97) with several important extensions. First, we employ treelet translation pairs instead of single word translations. Second, instead of modeling rearrangements as either preserving source order or swapping source order, we allow the dependents of a node to be ordered in any arbitrary manner and use the order model described in section 2.4 to estimate probabilities. Finally, we use a log-linear framework for model combination that allows any amount of other information to be modeled.

We will initially approach the decoding problem as a bottom up, exhaustive search. We define the set of all possible treelet translation pairs of the subtree rooted at each input node in the following manner: A treelet translation pair *x* is said to *match* the input dependency tree *S* iff



(a) Example input dependency tree.



(b) Example treelet translation pair.

Figure 4. Example decoder structures.

there is some connected subgraph *S'* that is identical to the source side of *x*. We say that *x* *covers* all the nodes in *S'* and is *rooted* at source node *s*, where *s* is the root of matched subgraph *S'*.

We first find all treelet translation pairs that match the input dependency tree. Each matched pair is placed on a list associated with the input node where the match is rooted. Moving bottom-up through the input dependency tree, we compute a list of *candidate translations* for the input subtree rooted at each node *s*, as follows:

Consider in turn each treelet translation pair *x* rooted at *s*. The treelet pair *x* may cover only a portion of the input subtree rooted at *s*. Find all descendents *s'* of *s* that are *not* covered by *x*, but whose parent *s''* is covered by *x*. At each such node *s''* look at all interleavings of the children of *s''* specified by *x*, if any, with each translation *t'* from the candidate translation list[5] of each child *s'*. Each such interleaving is scored using the models previously described and added to the candidate translation list for that input node. The resultant translation is the best scoring candidate for the root input node.

As an example, see the example dependency tree in Figure 4a and treelet translation pair in 4b. This treelet translation pair covers all the nodes in 4a except the subtrees rooted at *software* and *is*.

[5] Computed by the previous application of this procedure to *s'* during the bottom-up traversal.

We first compute (and cache) the candidate translation lists for the subtrees rooted at *software* and *is*, then construct full translation candidates by attaching those subtree translations to *installés* in all possible ways. The order of *sur* relative to *installés* is fixed; it remains to place the translated subtrees for *the software* and *is*. Note that if c is the count of children specified in the mapping and r is the count of subtrees translated via recursive calls, then there are $(c+r+1)!/(c+1)!$ orderings. Thus $(1+2+1)!/(1+1)! = 12$ candidate translations are produced for each combination of translations of *the software* and *is*.

3.1. Optimality-preserving optimizations

Dynamic Programming

Converting this exhaustive search to dynamic programming relies on the observation that scoring a translation candidate at a node depends on the following information from its descendents: the order model requires features from the root of a translated subtree, and the target language model is affected by the first and last two words in each subtree. Therefore, we need to keep the best scoring translation candidate for a given subtree for each combination of (head, leading bigram, trailing bigram), which is, in the worst case, $O(V^5)$, where V is the vocabulary size. The dynamic programming approach therefore does not allow for great savings in practice because a trigram target language model forces consideration of context external to each subtree.

Duplicate elimination

To eliminate unnecessary ordering operations, we first check that a given set of words has not been previously ordered by the decoder. We use an order-independent hash table where two trees are considered equal if they have the same tree structure and lexical choices after sorting each child list into a canonical order. A simpler alternate approach would be to compare bags-of-words. However since our possible orderings are bound by the induced tree structure, we might overzealously prune a candidate with a different tree structure that allows a better target order.

3.2. Lossy optimizations

The following optimizations do not preserve optimality, but work well in practice.

N-best lists

Instead of keeping the full list of translation candidates for a given input node, we keep a top-scoring subset of the candidates. While the decoder is no longer guaranteed to find the optimal translation, in practice the quality impact is minimal with a list size ≥ 10 (see Table 5.6).

Variable-sized n-best lists: A further speedup can be obtained by noting that the number of translations using a given treelet pair is exponential in the number of subtrees of the input not covered by that pair. To limit this explosion we vary the size of the *n*-best list on any recursive call in inverse proportion to the number of subtrees uncovered by the current treelet. This has the intuitive appeal of allowing a more thorough exploration of large treelet translation pairs (that are likely to result in better translations) than of smaller, less promising pairs.

Pruning treelet translation pairs

Channel model scores and treelet size are powerful predictors of translation quality. Heuristically pruning low scoring treelet translation pairs before the search starts allows the decoder to focus on combinations and orderings of high quality treelet pairs.

- Only keep those treelet translation pairs with an MLE probability above a threshold t.
- Given a set of treelet translation pairs with identical sources, keep those with an MLE probability within a ratio r of the best pair.
- At each input node, keep only the top k treelet translation pairs rooted at that node, as ranked first by size, then by MLE channel model score, then by Model 1 score. The impact of this optimization is explored in Table 5.6.

Greedy ordering

The complexity of the ordering step at each node grows with the factorial of the number of children to be ordered. This can be tamed by noting that given a fixed pre- and post-modifier count, our order model is capable of evaluating a single ordering decision independently from other ordering decisions.

One version of the decoder takes advantage of this to severely limit the number of ordering possibilities considered. Instead of considering all interleavings, it considers each potential modifier position in turn, greedily picking the most

		English	French
Training	Sentences	570,562	
	Words	7,327,251	8,415,882
	Vocabulary	72,440	80,758
	Singletons	38,037	39,496
Test	Sentences	10,000	
	Words	133,402	153,701

Table 4.1 Data characteristics

probable child for that slot, moving on to the next slot, picking the most probable among the remaining children for that slot and so on.

The complexity of greedy ordering is linear, but at the cost of a noticeable drop in BLEU score (see Table 5.4). Under default settings our system tries to decode a sentence with exhaustive ordering until a specified timeout, at which point it falls back to greedy ordering.

4. Experiments

We evaluated the translation quality of the system using the BLEU metric (Papineni et al., 02) under a variety of configurations. We compared against two radically different types of systems to demonstrate the competitiveness of this approach:

- Pharaoh: A leading phrasal SMT decoder (Koehn et al., 03).
- The MSR-MT system described in Section 1, an EBMT/hybrid MT system.

4.1. Data

We used a parallel English-French corpus containing 1.5 million sentences of Microsoft technical data (e.g., support articles, product documentation). We selected a cleaner subset of this data by eliminating sentences with XML or HTML tags as well as very long (>160 characters) and very short (<40 characters) sentences. We held out 2,000 sentences for development testing and parameter tuning, 10,000 sentences for testing, and 250 sentences for lambda training. We ran experiments on subsets of the training data ranging from 1,000 to 300,000 sentences. Table 4.1 presents details about this dataset.

4.2. Training

We parsed the source (English) side of the corpus using NLPWIN, a broad-coverage rule-based parser developed at Microsoft Research able to produce syntactic analyses at varying levels of depth (Heidorn, 02). For the purposes of these experiments we used a dependency tree output with part-of-speech tags and unstemmed surface words.

For word alignment, we used GIZA++, following a standard training regimen of five iterations of Model 1, five iterations of the HMM Model, and five iterations of Model 4, in both directions.

We then projected the dependency trees and used the aligned dependency tree pairs to extract treelet translation pairs and train the order model as described above. The target language model was trained using only the French side of the corpus; additional data may improve its performance. Finally we trained lambdas via Maximum BLEU (Och, 03) on 250 held-out sentences with a single reference translation, and tuned the decoder optimization parameters (*n*-best list size, timeouts etc) on the development test set.

Pharaoh

The same GIZA++ alignments as above were used in the Pharaoh decoder. We used the heuristic combination described in (Och & Ney, 03) and extracted phrasal translation pairs from this combined alignment as described in (Koehn et al., 03). Except for the order model (Pharaoh uses its own ordering approach), the same models were used: MLE channel model, Model 1 channel model, target language model, phrase count, and word count. Lambdas were trained in the same manner (Och, 03).

MSR-MT

MSR-MT used its own word alignment approach as described in (Menezes & Richardson, 01) on the same training data. MSR-MT does not use lambdas or a target language model.

5. Results

We present BLEU scores on an unseen 10,000 sentence test set using a single reference translation for each sentence. Speed numbers are the end-to-end translation speed in sentences per minute. All results are based on a training set size of 100,000 sentences and a phrase size of 4, except Table 5.2 which varies the phrase size and Table 5.3 which varies the training set size.

Results for our system and the comparison systems are presented in Table 5.1. Pharaoh monotone refers to Pharaoh with phrase reordering disabled. The difference between Pharaoh and the Treelet system is significant at the 99% confidence level under a two-tailed paired t-test.

	BLEU Score	Sents/min
Pharaoh monotone	37.06	4286
Pharaoh	38.83	162
MSR-MT	35.26	453
Treelet	40.66	10.1

Table 5.1 System comparisons

Table 5.2 compares Pharaoh and the Treelet system at different phrase sizes. While all the differences are statistically significant at the 99% confidence level, the wide gap at smaller phrase sizes is particularly striking. We infer that whereas Pharaoh depends heavily on long phrases to encapsulate reordering, our dependency tree-based ordering model enables credible performance even with single-word 'phrases'. We conjecture that in a language pair with large-scale ordering differences, such as English-Japanese, even long phrases are unlikely to capture the necessary reorderings, whereas our tree-based ordering model may prove more robust.

Max. size	Treelet BLEU	Pharaoh BLEU
1	37.50	23.18
2	39.84	32.07
3	40.36	37.09
4 (default)	40.66	38.83
5	40.71	39.41
6	40.74	39.72

Table 5.2 Effect of maximum treelet/phrase size

Table 5.3 compares the same systems at different training corpus sizes. All of the differences are statistically significant at the 99% confidence level. Noting that the gap widens at smaller corpus sizes, we suggest that our tree-based approach is more suitable than string-based phrasal SMT when translating from English into languages or domains with limited parallel data.

We also ran experiments varying different system parameters. Table 5.4 explores different ordering strategies, Table 5.5 looks at the impact of discontiguous phrases and Table 5.6 looks at the impact of decoder optimizations such as treelet pruning and *n*-best list size.

Ordering strategy	BLEU	Sents/min
No order model (monotone)	35.35	39.7
Greedy ordering	38.85	13.1
Exhaustive (default)	40.66	10.1

Table 5.4 Effect of ordering strategies

	BLEU Score	Sents/min
Contiguous only	40.08	11.0
Allow discontiguous	40.66	10.1

Table 5.5 Effect of allowing treelets that correspond to discontiguous phrases

	BLEU Score	Sents/min
Pruning treelets		
Keep top 1	28.58	144.9
… top 3	39.10	21.2
… top 5	40.29	14.6
… top 10 (default)	40.66	10.1
… top 20	40.70	3.5
Keep all	40.29	3.2
N-best list size		
1-best	37.28	175.4
5-best	39.96	79.4
10-best	40.42	23.3
20-best (default)	40.66	10.1
50-best	39.39	3.7

Table 5.6 Effect of optimizations

6. Discussion

We presented a novel approach to syntactically-informed statistical machine translation that leverages a parsed dependency tree representation of the source language via a tree-based ordering model and treelet phrase extraction. We showed that it significantly outperforms a leading phrasal SMT system over a wide range of training set sizes and phrase sizes.

Constituents vs. dependencies: Most attempts at

	1k	3k	10k	30k	100k	300k
Pharaoh	17.20	22.51	27.70	33.73	38.83	42.75
Treelet	18.70	25.39	30.96	35.81	40.66	44.32

Table 5.3 Effect of training set size on treelet translation and comparison system

syntactic SMT have relied on a constituency analysis rather than dependency analysis. While this is a natural starting point due to its well-understood nature and commonly available tools, we feel that this is not the most effective representation for syntax in MT. Dependency analysis, in contrast to constituency analysis, tends to bring semantically related elements together (e.g., verbs become adjacent to all their arguments) and is better suited to lexicalized models, such as the ones presented in this paper.

7. Future work

The most important contribution of our system is a linguistically motivated ordering approach based on the source dependency tree, yet this paper only explores one possible model. Different model structures, machine learning techniques, and target feature representations all have the potential for significant improvements.

Currently we only consider the top parse of an input sentence. One means of considering alternate possibilities is to build a packed forest of dependency trees and use this in decoding translations of each input sentence.

As noted above, our approach shows particular promise for language pairs such as English-Japanese that exhibit large-scale reordering and have proven difficult for string-based approaches. Further experimentation with such language pairs is necessary to confirm this. Our experience has been that the quality of GIZA++ alignments for such language pairs is inadequate. Following up on ideas introduced by (Cherry & Lin, 03) we plan to explore ways to leverage the dependency tree to improve alignment quality.

References

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

Aue, Anthony, Arul Menezes, Robert C. Moore, Chris Quirk, and Eric Ringger. Statistical machine translation using labeled semantic dependency graphs. TMI 2004.

Charniak, Eugene, Kevin Knight, and Kenji Yamada. Syntax-based language models for statistical machine translation. MT Summit 2003.

Cherry, Colin and Dekang Lin. A probability model to improve word alignment. ACL 2003.

Chickering, David Maxwell. The WinMine Toolkit. Microsoft Research Technical Report: MSR-TR-2002-103.

Ding, Yuan and Martha Palmer. Automatic learning of parallel dependency treelet pairs. IJCNLP 2004.

Heidorn, George. (2000). “Intelligent writing assistance”. In Dale et al. Handbook of Natural Language Processing, Marcel Dekker.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. Statistical phrase based translation. NAACL 2003.

Lin, Dekang. A path-based transfer model for machine translation. COLING 2004.

Menezes, Arul and Stephen D. Richardson. A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora. DDMT Workshop, ACL 2001.

Och, Franz Josef and Hermann Ney. A systematic comparison of various statistical alignment models, Computational Linguistics, 29(1):19-51, 2003.

Och, Franz Josef. Minimum error rate training in statistical machine translation. ACL 2003.

Och, Franz Josef, et al. A smorgasbord of features for statistical machine translation. HLT/NAACL 2004.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. ACL 2002.

Quirk, Chris, Arul Menezes, and Colin Cherry. Dependency Tree Translation. Microsoft Research Technical Report: MSR-TR-2004-113.

Ringger, Eric, et al. Linguistically informed statistical models of constituent structure for ordering in sentence realization. COLING 2004.

Thurmair, Gregor. Comparing rule-based and statistical MT output. Workshop on the amazing utility of parallel and comparable corpora, LREC, 2004.

Vogel, Stephan, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. The CMU statistical machine translation system. MT Summit 2003.

Wu, Dekai. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. Computational Linguistics, 23(3):377–403, 1997.

Yamada, Kenji and Kevin Knight. A syntax-based statistical translation model. ACL, 2001.

QARLA:A Framework for the Evaluation of Text Summarization Systems

Enrique Amigó, Julio Gonzalo, Anselmo Peñas, Felisa Verdejo
Departamento de Lenguajes y Sistemas Informáticos
Universidad Nacional de Educación a Distancia
c/Juan del Rosal, 16 - 28040 Madrid - Spain
{enrique,julio,anselmo,felisa}@lsi.uned.es

Abstract

This paper presents a probabilistic framework, QARLA, for the evaluation of text summarisation systems. The input of the framework is a set of manual (reference) summaries, a set of baseline (automatic) summaries and a set of similarity metrics between summaries. It provides i) a measure to evaluate the quality of any set of similarity metrics, ii) a measure to evaluate the quality of a summary using an optimal set of similarity metrics, and iii) a measure to evaluate whether the set of baseline summaries is reliable or may produce biased results.

Compared to previous approaches, our framework is able to combine different metrics and evaluate the quality of a set of metrics without any a-priori weighting of their relative importance. We provide quantitative evidence about the effectiveness of the approach to improve the automatic evaluation of text summarisation systems by combining several similarity metrics.

1 Introduction

The quality of an automatic summary can be established mainly with two approaches:

Human assessments: The output of a number of summarisation systems is compared by human judges, using some set of evaluation guidelines.

Proximity to a gold standard: The best automatic summary is the one that is closest to some reference summary made by humans.

Using human assessments has some clear advantages: the results of the evaluation are interpretable, and we can trace what a system is doing well, and what is doing poorly. But it also has a couple of serious drawbacks: i) different human assessors reach different conclusions, and ii) the outcome of a comparative evaluation exercise is not directly reusable for new techniques, i.e., a summarisation strategy developed after the comparative exercise cannot be evaluated without additional human assessments made from scratch.

Proximity to a gold standard, on the other hand, is a criterion that can be automated (see Section 6), with the advantages of i) being objective, and ii) once gold standard summaries are built for a comparative evaluation of systems, the resulting testbed can iteratively be used to refine text summarisation techniques and re-evaluate them automatically.

This second approach, however, requires solving a number of non-trivial issues. For instance, (i) How can we know whether an evaluation metric is good enough for automatic evaluation?, (ii) different users produce different summaries, all of them equally good as gold standards, (iii) if we have several metrics which test different features of a summary, how can we combine them into an optimal test?, (iv) how do we know if our test bed

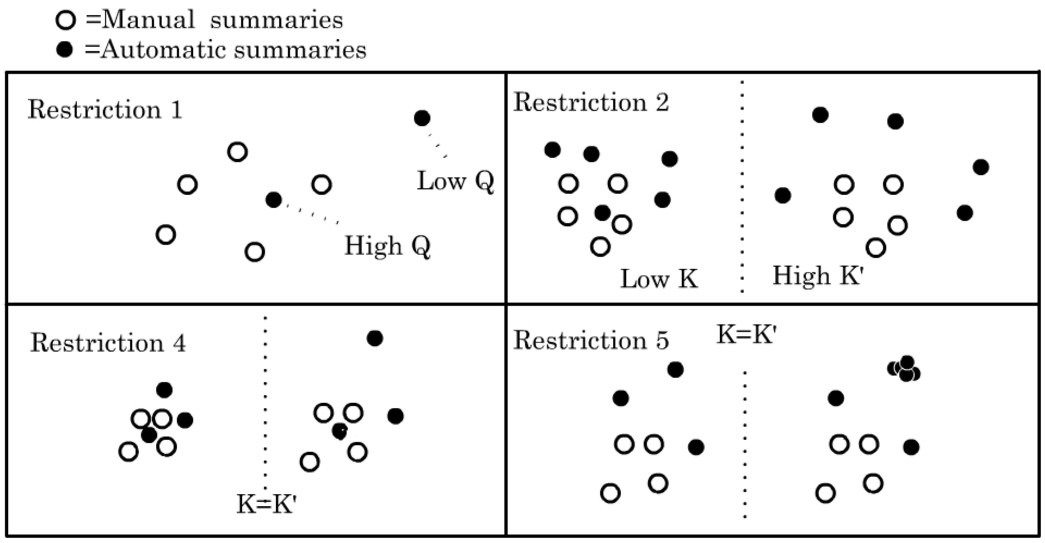


Figure 1: Illustration of some of the restrictions on Q, K

is reliable, or the evaluation outcome may change by adding, for instance, additional gold standards?

In this paper, we introduce a probabilistic framework, QARLA, that addresses such issues. Given a set of manual summaries and another set of baseline summaries per task, together with a set of similarity metrics, QARLA provides quantitative measures to (i) select and combine the best (independent) metrics (KING measure), (ii) apply the best set of metrics to evaluate automatic summaries (QUEEN measure), and (iii) test whether evaluating with that test-bed is reliable (JACK measure).

2 Formal constraints on any evaluation framework based on similarity metrics

We are looking for a framework to evaluate automatic summarisation systems objectively using similarity metrics to compare summaries. The input of the framework is:

- A summarisation task (e.g. topic oriented, informative multi-document summarisation on a given domain/corpus).
- A set T of test cases (e.g. topic/document set pairs for the example above)
- A set of summaries M produced by humans (*models*), and a set of automatic summaries A (*peers*), for every test case.
- A set X of similarity metrics to compare summaries.

An evaluation framework should include, at least:

- A measure $Q_{M,X}(a) \in [0,1]$ that estimates the quality of an automatic summary a, using the similarity metrics in X to compare the summary with the models in M. With Q, we can compare the quality of automatic summaries.
- A measure $K_{M,A}(X) \in [0,1]$ that estimates the suitability of a set of similarity metrics X for our evaluation purposes. With K, we can choose the best similarity metrics.

Our main assumption is that all manual summaries are equally optimal and, while they are likely to be different, the best similarity metric is the one that identifies and uses the features that are common to all manual summaries, grouping and separating them from the automatic summaries.

With these assumption in mind, it is useful to think of some formal restrictions that any evaluation framework Q, K must hold. We will consider the following ones (see illustrations in Figure 1):

(1) Given two automatic summaries a, a' and a similarity measure x, if a is more distant to all manual summaries than a', then a cannot be better

than a'. Formally: $\forall m \in M.x(a,m) < x(a',m) \rightarrow Q_{M,x}(a) \leq Q_{M,x}(a')$
(2) A similarity metric x is better when it is able to group manual summaries more closely, while keeping them more distant from automatic summaries: $(\forall m, m' \in M.x(m,m') > x'(m,m') \wedge \forall m \in M, a \in Ax(a,m) < x'(a,m)) \rightarrow K_{M,A}(x) > K_{M,A}(x')$
(3) If x is a perfect similarity metric, the quality of a manual summary cannot be zero: $K_{M,A}(x) = 1 \rightarrow \forall m \in M.Q_{M,x}(m) > 0$
(4) The quality of a similarity metric or a summary should not be dependent on scale issues. In general, if $x' = f(x)$ with f being a growing monotonic function, then $K_{M,A}(x) = K_{M,A}(x')$ and $Q_{M,x}(a) = Q_{M,x'}(a)$.
(5) The quality of a similarity metric should not be sensitive to repeated elements in A, i.e. $K_{M,A\cup\{a\}}(x) = K_{M,A\cup\{a,a\}}(x)$.
(6) A random metric x should have $K_{M,A}(x) = 0$.
(7) A non-informative (constant) metric x should have $K_{M,A}(x) = 0$.

3 QARLA evaluation framework

3.1 QUEEN: Estimation of the quality of an automatic summary

We are now looking for a function $Q_{M,x}(a)$ that estimates the quality of an automatic summary $a \in A$, given a set of models M and a similarity metric x.

An obvious first attempt would be to compute the average similarity of a to all model summaries in M in a test sample. But such a measure depends on scale properties: metrics producing larger similarity values will produce larger Q values; and, depending on the scale properties of x, this cannot be solved just by scaling the final Q value.

A probabilistic measure that solves this problem and satisfies all the stated formal constraints is:

$$\text{QUEEN}_{x,M}(a) \equiv P(x(a,m) \geq x(m',m''))$$

which defines the quality of an automatic summary a as the probability over triples of manual summaries m, m', m'' that a is closer to a model than the other two models to each other. This measure draws from the way in which some formal restrictions on Q are stated (by comparing similarity values), and is inspired in the QARLA criterion introduced in (Amigo et al., 2004).

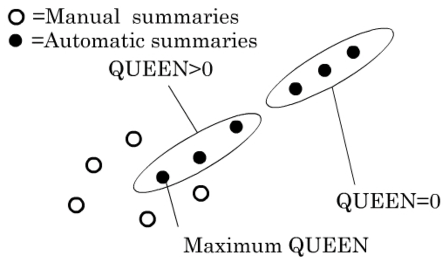


Figure 2: Summaries quality in a similarity metric space

Figure 2 illustrates some of the features of the QUEEN estimation:

- Peers which are very far from the set of models all receive QUEEN $= 0$. In other words, QUEEN does not distinguish between very poor automatic summarisation strategies. While this feature reduces granularity of the ranking produced by QUEEN, we find it desirable, because in such situations, the values returned by a similarity measure are probably meaningless.

- The value of QUEEN is maximised for the peers that “merge” with the models. For QUEEN values between 0.5 and 1, peers are effectively merged with the models.

- An ideal metric (that puts all models together) would give QUEEN$(m) = 1$ for all models, and QUEEN$(a) = 0$ for all peers that are not put together with the models. This is a reasonable boundary condition saying that, if we can distinguish between models and peers perfectly, then all peers are poor emulations of human summarising behaviour.

3.2 Generalisation of QUEEN to metric sets

It is desirable, however, to have the possibility of evaluating summaries with respect to several metrics together. Let us imagine, for instance, that the best metric turns out to be a ROUGE (Lin and Hovy, 2003a) variant that only considers unigrams to compute similarity. Now consider a summary which has almost the same vocabulary as a human summary, but with a random scrambling of the words which makes it unreadable. Even if the unigram measure is the best hint of similarity to human performance, in this case it would produce a high similarity value, while any measure based on 2-grams, 3-grams or on any simple syntactic property would detect that the summary is useless.

The issue is, therefore, how to find informative metrics, and then how to combine them into an optimal single quality estimation for automatic summaries. The most immediate way of combining metrics is via some weighted linear combination. But our example suggests that this is not the optimal way: the unigram measure would take the higher weight, and therefore it would assign a fair amount of credit to a summary that can be strongly rejected with other criteria.

Alternatively, we can assume that a summary is better if it is closer to the model summaries according to all metrics. We can formalise this idea by introducing a universal quantifier on the variable x in the QUEEN formula. In other words, $\text{QUEEN}_{X,M}(a)$ can be defined as the probability, measured over $M \times M \times M$, that for every metric in X the automatic summary a is closer to a model than two models to each other.

$$\text{QUEEN}_{X,M}(a) \equiv P(\forall x \in X . x(a,m) \geq x(m',m''))$$

We can think of the generalised QUEEN measure as a way of using a set of tests (every similarity metric in X) to falsify the hypothesis that a given summary a is a model. If, for every comparison of similarities between a, m, m', m'', there is at least one test that a does not pass, then a is rejected as a model.

This generalised measure is not affected by the scale properties of every individual metric, i.e. it does not require metric normalisation and it is not affected by metric weighting. In addition, it still satisfies the properties enumerated for its single-metric counterpart.

Of course, the quality ranking provided by QUEEN is meaningless if the similarity metric x does not capture the essential features of the models. Therefore, we need to estimate the quality of similarity metrics in order to use QUEEN effectively.

3.3 KING: estimation of the quality of a similarity metric

Now we need a measure $K_{M,A}(x)$ that estimates the quality of a similarity metric x to evaluate automatic summaries (peers) by comparison to human-produced models.

In order to build a suitable K estimation, we will again start from the hypothesis that the best metric is the one that best characterises human summaries as opposed to automatic summaries. Such a metric should identify human summaries as closer to each other, and more distant to peers (second constraint in Section 2). By analogy with QUEEN, we can try (for a single metric):

$$K_{M,A}(x) \equiv P(x(a,m) < x(m',m'')) = \\ 1 - \overline{(\text{QUEEN}_{x,M}(a))}$$

which is the probability that two models are closer to each other than a third model to a peer, and has smaller values when the average QUEEN value of peers decreases. The generalisation of K to metric sets would be simply:

$$K_{M,A}(X) \equiv 1 - \overline{(\text{QUEEN}_{X,M}(a)))}$$

This measure, however, does not satisfy formal conditions 3 and 5. Condition 3 is violated because, given a limited set of models, the K measure grows with a large number of metrics in X, eventually reaching $K = 1$ (perfect metric set). But in this situation, QUEEN(m) becomes 0 for all models, because there will always exist a metric that breaks the universal quantifier condition over x.

We have to look, then, for an alternative formulation for K. The best K should minimise QUEEN(a), but having the quality of the models as a reference. A direct formulation can be:

$$K_{M,A}(X) = P(\text{QUEEN}(m) > \text{QUEEN}(a))$$

According to this formula, the quality of a metric set X is the probability that the quality of a model is higher than the quality of a peer according to this metric set. This formula satisfies all formal conditions except 5 ($K_{M,A\cup\{a\}}(x) = K_{M,A\cup\{a,a\}}(x)$), because it is sensitive to repeated peers. If we add a large set of identical (or very similar peers), K will be biased towards this set.

We can define a suitable K that satisfies condition 5 if we apply a universal quantifier on a. This is what we call the KING measure:

$$\text{KING}_{M,A}(X) \equiv$$
$$P(\forall a \in A.\text{QUEEN}_{M,X}(m) > \text{QUEEN}_{M,X}(a))$$

KING is the probability that a model is better than any peer in a test sample. In terms of a quality ranking, it is the probability that a model gets a better ranking than all peers in a test sample. Note that KING satisfies all restrictions because it uses QUEEN as a quality estimation for summaries; if QUEEN is substituted for a different quality measure, some of the properties might not hold any longer.

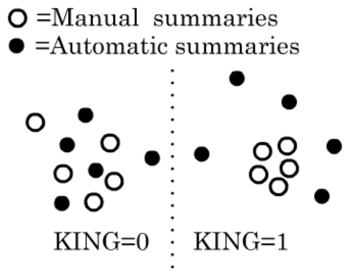


Figure 3: Metrics quality representation

Figure 3 illustrates the behaviour of the KING measure in boundary conditions. The leftmost figure represents a similarity metric which mixes models and peers randomly. Therefore, $P(\text{QUEEN}(m) > \text{QUEEN}(a)) \approx 0.5$. As there are seven automatic summaries, $\text{KING} = P(\forall a \in A, \text{QUEEN}(m) > \text{QUEEN}(a)) \approx 0.5^7 \approx 0$

The rightmost figure represents a metric which is able to group models and separate them from peers. In this case, $\text{QUEEN}(a) = 0$ for all peers, and then $\text{KING}(x) = 1$.

3.4 JACK:Reliability of the peers set

Once we detect a difference in quality between two summarisation systems, the question is now whether this result is reliable. Would we get the same results using a different test set (different examples, different human summarisers (models) or different baseline systems)?

The first step is obviously to apply statistical significance tests to the results. But even if they give a positive result, it might be insufficient. The problem is that the estimation of the probabilities in $\text{KING}, \text{QUEEN}$ assumes that the sample sets M, A are not biased. If M, A are biased, the results can be statistically significant and yet unreliable. The set of examples and the behaviour of human summarisers (models) should be somehow controlled either for homogeneity (if the intended profile of examples and/or users is narrow) or representativity (if it is wide). But how to know whether the set of automatic summaries is representative and therefore is not penalising certain automatic summarisation strategies?

Our goal is, therefore, to have some estimation $\text{JACK}(X, M, A)$ of the reliability of the test set to compute reliable $\text{QUEEN}, \text{KING}$ measures. We can think of three reasonable criteria for this estimation:

1. All other things being equal, if the elements of A are more heterogeneous, we are enhancing the representativeness of A (we have a more diverse set of (independent) automatic summarization strategies represented), and therefore the reliability of the results should be higher. Reversely, if all automatic summarisers employ similar strategies, we may end up with a biased set of peers.

2. All other things being equal, if the elements of A are closer to the model summaries in M, the reliability of the results should be higher.

3. Adding items to A should not reduce its reliability.

A possible formulation for JACK which satisfies that criteria is:

$$\text{JACK}(X, M, A) \equiv P(\exists a, a' \in A.\text{QUEEN}(a) >$$
$$0 \wedge \text{QUEEN}(a') > 0 \wedge \forall x \in X.x(a, a') \leq x(a, m))$$

i.e. the probability over all model summaries m of finding a couple of automatic summaries a, a'

which are closer to each other than to m according to all metrics.

This measure satisfies all three constraints: it can be enlarged by increasing the similarity of the peers to the models (the $x(m, a)$ factor in the inequality) or decreasing the similarity between automatic summaries (the $x(a, a')$ factor in the inequality). Finally, adding elements to A can only increase the chances of finding a pair of automatic summaries satisfying the condition in JACK.

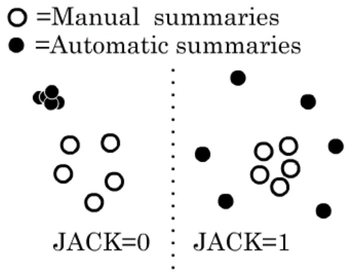


Figure 4: JACK values

Figure 4 illustrates how JACK works: in the leftmost part of the figure, peers are grouped together and far from the models, giving a low JACK value. In the rightmost part of the figure, peers are distributed around the set of models, closely surrounding them, receiving a high JACK value.

4 A Case of Study

In order to test the behaviour of our evaluation framework, we have applied it to the ISCORPUS described in (Amigo et al., 2004). The ISCORPUS was built to study an *Information Synthesis* task, where a (large) set of relevant documents has to be studied to give a brief, well-organised answer to a complex need for information. This corpus comprises:

- Eight topics extracted from the CLEF Spanish Information Retrieval test set, slightly reworded to move from a document retrieval task (find documents about hunger strikes in...) into an Information Synthesis task (make a report about major causes of hunger strikes in...).
- One hundred relevant documents per topic taken from the CLEF EFE 1994 Spanish newswire collection.
- M: Manual extractive summaries for every topic made by 9 different users, with a 50-sentence upper limit (half the number of relevant documents).
- A: 30 automatic reports for every topic made with baseline strategies. The 10 reports with highest sentence overlap with the manual summaries were selected as a way to increase the quality of the baseline set.

We have considered the following similarity metrics:

ROUGESim: ROUGE is a standard measure to evaluate summarisation systems based on n-gram recall. We have used ROUGE-1 (only unigrams with lemmatization and stop word removal), which gives good results with standard summaries (Lin and Hovy, 2003a). ROUGE can be turned into a similarity metric *ROUGESim* simply by considering only one model when computing its value.

SentencePrecision: Given a reference and a contrastive summary, the number of fragments of the contrastive summary which are also in the reference summary, in relation to the size of the reference summary.

SentenceRecall: Given a reference and a contrastive summary, the number of fragments of the reference summary which are also in the contrastive summary, in relation to the size of the contrastive summary.

DocSim: The number of documents used to select fragments in both summaries, in relation to the size of the contrastive summary.

VectModelSim: Derived from the Euclidean distance between vectors of relative word frequencies representing both summaries.

NICOS (key concept overlap): Same as *VectModelSim*, but using key-concepts (manually identified by the human summarisers after producing the summary) instead of all non-empty words.

TruncatedVectModel$_n$: Same as *VectModelSim*, but using only the n more frequent terms in the reference summary. We have used 10 variants of this measure with $n = 1, 8, 64, 512$.

4.1 Quality of Similarity Metric Sets

Figure 5 shows the quality (KING values averaged over the eight ISCORPUS topics) of every individual metric. The rightmost part of the figure also shows the quality of two metric sets:

- The first one ($\{$*ROUGESim, VectModelSim, TruncVectModel.1*$\}$) is the metric set that maximises KING, using only similarity metrics that do not require manual annotation (i.e. excluding *NICOS*) or can only be applied to extractive summaries (i.e. *DocSim*, *SentenceRecall* and *SentencePrecision*).

- The second one ($\{$ *TruncVectModel.1, ROUGESim, DocSim, VectModelSim* $\}$) is the best combination considering all metrics.

The best result of individual metrics is obtained by *ROUGESim* (0.39). All other individual metrics give scores below 0.31. Both metric sets, on the other, are better than *ROUGESim* alone, confirming that metric combination is feasible to improve system evaluation. The quality of the best metric set (0.47) is 21% better than *ROUGESim*.

4.2 Reliability of the test set

The 30 automatic summaries (baselines) per topic were built with four different classes of strategies: i) picking up the first sentence from assorted subsets of documents, ii) picking up first and second sentences from assorted documents, iii) picking up first, second or third sentences from assorted documents, and iv) picking up whole documents with different algorithms to determine which are the most representative documents.

Figure 6 shows the reliability (JACK) of every subset, and the reliability of the whole set of automatic summaries, computed with the best metric set. Note that the individual subsets are all below 0.2, while the reliability of the full set of peers goes up to 0.57. That means that the condition in JACK is satisfied for more than half of the models. This value would probably be higher if state-of-the-art summarisation techniques were represented in the set of peers.

5 Testing the predictive power of the framework

The QARLA probabilistic framework is designed to evaluate automatic summarisation systems and, at the same time, similarity metrics conceived as well to evaluate summarisation systems. Therefore, testing the validity of the QARLA proposal implies some kind of meta-meta-evaluation, something which seems difficult to design or even to define.

It is relatively simple, however, to perform some simple cross-checkings on the ISCORPUS data to verify that the qualitative information described above is reasonable. This is the test we have implemented:

If we remove a model m from M, and pretend it is the output of an automatic summariser, we can evaluate the peers set A and the new peer m using $M' = M\backslash\{m\}$ as the new model set. If the evaluation metric is good, the quality of the new peer m should be superior to all other peers in A. What we have to check, then, is whether the average quality of a human summariser on all test cases (8 topics in ISCORPUS) is superior to the average quality of any automatic summariser. We have 9 human subjects in the ISCORPUS test bed; therefore, we can repeat this test nine times.

With this criterion, we can compare our quality measure Q with state-of-the-art evaluation measures such as ROUGE variants. Table 1 shows the results of applying this test on ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-4 (as state-of-the-art references) and QUEEN$($*ROUGESim*$)$, QUEEN$($Best Metric Combination$)$ as representatives of the QARLA framework. Even if the test is very limited by the number of topics, it confirms the potential of the framework, with the highest KING metric combination doubling the performance of the best ROUGE measure (6/9 versus 3/9 correct detections).

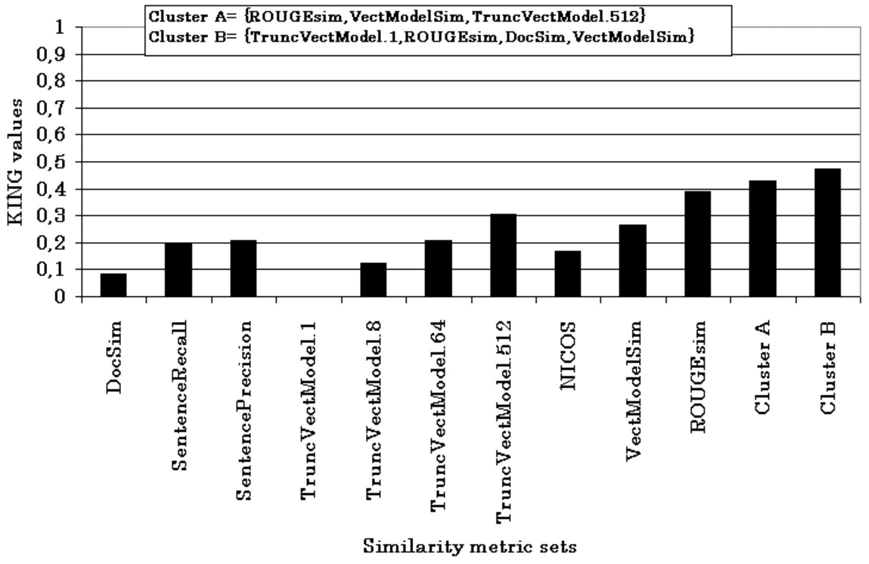


Figure 5: Quality of similarity metrics

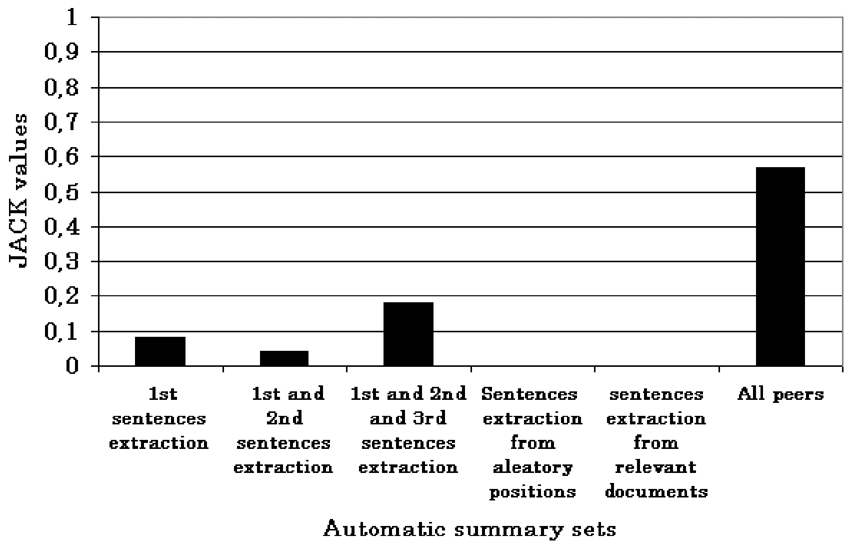


Figure 6: Reliability of ISCORPUS peer sets

Evaluation criterion	human summarisers ranked first
ROUGE-1	3/9
ROUGE-2	2/9
ROUGE-3	1/9
ROUGE-4	1/9
QUEEN(ROUGESim)	4/9
QUEEN(Best Metric Combination)	6/9

Table 1: Results of the test of identifying the manual summariser

6 Related work and discussion

6.1 Application of similarity metrics to evaluate summaries

Both in Text Summarisation and Machine Translation, the automatic evaluation of systems consists of computing some similarity metric between the system output and a human model summary. Systems are then ranked in order of decreasing similarity to the gold standard. When there are more than one reference items, similarity is calculated over a pseudo-summary extracted from every model. BLEU (Papineni et al., 2001) and ROUGE (Lin and Hovy, 2003a) are the standard similarity metrics used in Machine Translation and Text Summarisation. Generating a pseudo-summary from every model, the results of a evaluation metric might depend on the scale properties of the metric regarding different models; our QUEEN measure, however, does not depend on scales.

Another problem of the direct application of a single evaluation metric to rank systems is how to combine different metrics. The only way to do this is by designing an algebraic combination of the individual metrics into a new combined metric, i.e. by deciding the weight of each individual metric beforehand. In our framework, however, it is not necessary to prescribe how similarity metrics should be combined, not even to know which ones are individually better indicators.

6.2 Meta-evaluation of similarity metrics

The question of how to know which similarity metric is best to evaluate automatic summaries/translations has been addressed by

- comparing the quality of automatic items with the quality of manual references (Culy and Riehemann, 2003; Lin and Hovy, 2003b). If the metric does not identify that the manual references are better, then it is not good enough for evaluation purposes.
- measuring the correlation between the values given by different metrics (Coughlin, 2003).
- measuring the correlation between the rankings generated by each metric and rankings generated by human assessors. (Joseph P. Turian and Melamed, 2003; Lin and Hovy, 2003a).

The methodology which is closest to our framework is ORANGE (Lin, 2004), which evaluates a similarity metric using the average ranks obtained by reference items within a baseline set. As in our framework, ORANGE performs an automatic meta-evaluation, there is no need for human assessments, and it does not depend on the scale properties of the metric being evaluated (because changes of scale preserve rankings). The ORANGE approach is, indeed, closely related to the original QARLA measure introduced in (Amigo et al., 2004).

Our $\text{KING}, \text{QUEEN}, \text{JACK}$ framework, however, has a number of advantages over ORANGE:

- It is able to combine different metrics, and evaluate the quality of metric sets, without any a-priori weighting of their relative importance.
- It is not sensitive to repeated (or very similar) baseline elements.
- It provides a mechanism, JACK, to check whether a set X, M, A of metrics, manual and baseline items is reliable enough to produce a stable evaluation of automatic summarisation systems.

Probably the most significant improvement over ORANGE is the ability of $\text{KING}, \text{QUEEN}, \text{JACK}$ to combine automatically the information of different metrics. We believe that a comprehensive automatic evaluation of a summary must necessarily capture different aspects of the problem with different metrics, and that the results of every individual metric should not be combined in any prescribed algebraic way (such as a linear weighted combination). Our framework satisfies this condition. An advantage of $ORANGE$, however, is that it does not require a large number of gold standards to reach stability, as in the case of $QARLA$.

Finally, it is interesting to compare the rankings produced by $QARLA$ with the output of human assessments, even if the philosophy of $QARLA$ is not considering human assessments as the gold standard for evaluation. Our initial tests on DUC

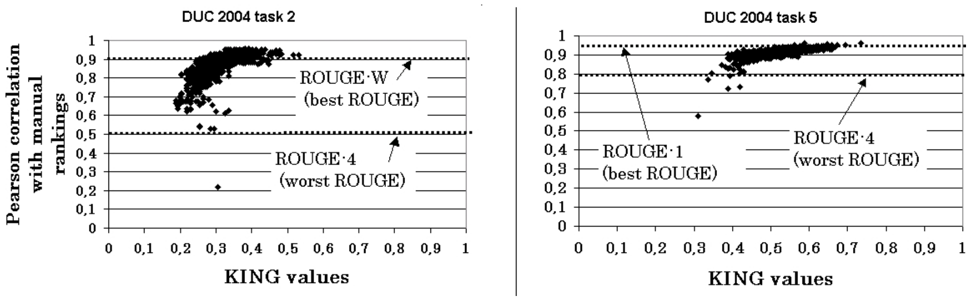


Figure 7: KING vs Pearson correlation with manual rankings in DUC for 1024 metrics combinations

test beds are very promising, reaching Pearson correlations of 0.9 and 0.95 between human assessments and QUEEN values for DUC 2004 tasks 2 and 5 (Over and Yen, 2004), using metric sets with highest KING values. The figure 7 shows how Pearson correlation grows up with higher KING values for 1024 metric combinations.

Acknowledgments

We are indebted to Ed Hovy, Donna Harman, Paul Over, Hoa Dang and Chin-Yew Lin for their inspiring and generous feedback at different stages in the development of QARLA. We are also indebted to NIST for hosting Enrique Amigó as a visitor and for providing the DUC test beds. This work has been partially supported by the Spanish government, project R2D2 (TIC-2003-7180).

References

E. Amigo, V. Peinado, J. Gonzalo, A. Peñas, and F. Verdejo. 2004. An empirical study of information synthesis task. In *Proceedings of the 42th Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, July.

Deborah Coughlin. 2003. Correlating Automated and Human Assessments of Machine Translation Quality. In *In Proceedings of MT Summit IX*, New Orleans,LA.

Christopher Culy and Susanne Riehemann. 2003. The Limits of N-Gram Translation Evaluation Metrics. In *Proceedings of MT Summit IX*, New Orleans,LA.

Luke Shen Joseph P. Turian and I. Dan Melamed. 2003. Evaluation of Machine Translation and its Evaluation. In *In Proceedings of MT Summit IX*, New Orleans,LA.

C. Lin and E. H. Hovy. 2003a. Automatic Evaluation of Summaries Using N-gram Co-ocurrence Statistics. In *Proceeding of 2003 Language Technology Conference (HLT-NAACL 2003)*.

Chin-Yew Lin and Eduard Hovy. 2003b. The Potential and Limitations of Automatic Sentence Extraction for Summarization. In Dragomir Radev and Simone Teufel, editors, *HLT-NAACL 2003 Workshop: Text Summarization (DUC03)*, Edmonton, Alberta, Canada, May 31 - June 1. Association for Computational Linguistics.

C. Lin. 2004. Orange: a Method for Evaluating Automatic Metrics for Machine Translation. In *Proceedings of the 36th Annual Conference on Computational Linguisticsion for Computational Linguistics (Coling'04)*, Geneva, August.

P. Over and J. Yen. 2004. An introduction to duc 2004 intrinsic evaluation of generic new text summarization systems. In *Proceedings of DUC 2004 Document Understanding Workshop, Boston*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, jul.

Supervised and Unsupervised Learning for Sentence Compression

Jenine Turner and Eugene Charniak
Department of Computer Science
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{jenine|ec}@cs.brown.edu

Abstract

In *Statistics-Based Summarization - Step One: Sentence Compression*, Knight and Marcu (Knight and Marcu, 2000) (K&M) present a noisy-channel model for sentence compression. The main difficulty in using this method is the lack of data; Knight and Marcu use a corpus of 1035 training sentences. More data is not easily available, so in addition to improving the original K&M noisy-channel model, we create unsupervised and semi-supervised models of the task. Finally, we point out problems with modeling the task in this way. They suggest areas for future research.

1 Introduction

Summarization in general, and sentence compression in particular, are popular topics. Knight and Marcu (henceforth K&M) introduce the task of statistical sentence compression in *Statistics-Based Summarization - Step One: Sentence Compression* (Knight and Marcu, 2000). The appeal of this problem is that it produces summarizations on a small scale. It simplifies general compression problems, such as text-to-abstract conversion, by eliminating the need for coherency between sentences. The model is further simplified by being constrained to word deletion: no rearranging of words takes place. Others have performed the sentence compression task using syntactic approaches to this problem (Mani et al., 1999) (Zajic et al., 2004), but we focus exclusively on the K&M formulation. Though the problem is simpler, it is still pertinent to current needs; generation of captions for television and audio scanning services for the blind (Grefenstette, 1998), as well as compressing chosen sentences for headline generation (Angheluta et al., 2004) are examples of uses for sentence compression. In addition to simplifying the task, K&M's noisy-channel formulation is also appealing.

In the following sections, we discuss the K&M noisy-channel model. We then present our cleaned up, and slightly improved noisy-channel model. We also develop unsupervised and semi-supervised (our term for a combination of supervised and unsupervised) methods of sentence compression with inspiration from the K&M model, and create additional constraints to improve the compressions. We conclude with the problems inherent in both models.

2 The Noisy-Channel Model

2.1 The K&M Model

The K&M probabilistic model, adapted from machine translation to this task, is the noisy-channel model. In machine translation, one imagines that a string was originally in English, but that someone adds some noise to make it a foreign string. Analogously, in the sentence compression model, the short string is the original sentence and someone adds noise, resulting in the longer sentence. Using this framework, the end goal is, given a long sentence l, to determine the short sentence s that maximizes

$P(s \mid l)$. By Bayes Rule,

$$P(s \mid l) = \frac{P(l \mid s)P(s)}{P(l)} \tag{1}$$

The probability of the long sentence, $P(l)$ can be ignored when finding the maximum, because the long sentence is the same in every case.

$P(s)$ is the source model: the probability that s is the original sentence. $P(l \mid s)$ is the channel model: the probability the long sentence is the expanded version of the short. This framework independently models the grammaticality of s (with $P(s)$) and whether s is a good compression of l ($P(l \mid s)$).

The K&M model uses parse trees for the sentences. These allow it to better determine the probability of the short sentence and to obtain alignments from the training data. In the K&M model, the sentence probability is determined by combining a probabilistic context free grammar (PCFG) with a word-bigram score. The joint rules used to create the compressions are generated by aligning the nodes of the short and long trees in the training data to determine expansion probabilities ($P(l \mid s)$).

Recall that the channel model tries to find the probability of the long string with respect to the short string. It obtains these probabilities by aligning nodes in the parsed parallel training corpus, and counting the nodes that align as "joint events." For example, there might be *S* → *NP VP PP* in the long sentence and *S* → *NP VP* in the short sentence; we count this as one joint event. Non-compressions, where the long version is the same as the short, are also counted. The expansion probability, as used in the channel model, is given by

$$P_{expand}(l \mid s) = \frac{count(joint(l, s))}{count(s)} \tag{2}$$

where $count(joint(l, s))$ is the count of alignments of the long rule and the short. Many compressions do not align exactly. Sometimes the parses do not match, and sometimes there are deletions that are too complex to be modeled in this way. In these cases sentence pairs, or sections of them, are ignored.

The K&M model creates a packed parse forest of all possible compressions that are grammatical with respect to the Penn Treebank (Marcus et al., 1993). Any compression given a zero expansion probability according to the training data is instead assigned a very small probability. A tree extractor (Langkilde, 2000) collects the short sentences with the highest score for $P(s \mid l)$.

2.2 Our Noisy-Channel Model

Our starting implementation is intended to follow the K&M model fairly closely. We use the same 1067 pairs of sentences from the Ziff-Davis corpus, with 32 used as testing and the rest as training. The main difference between their model and ours is that instead of using the rather ad-hoc K&M language model, we substitute the syntax-based language model described in (Charniak, 2001).

We slightly modify the channel model equation to be $P(l \mid s) = P_{expand}(l \mid s)P_{deleted}$, where $P_{deleted}$ is the probability of adding the deleted subtrees back into s to get l. We determine this probability also using the Charniak language model.

We require an extra parameter to encourage compression. We create a development corpus of 25 sentences from the training data in order to adjust this parameter. That we require a parameter to encourage compression is odd as K&M required a parameter to discourage compression, but we address this point in the penultimate section.

Another difference is that we only generate short versions for which we have rules. If we have never before seen the long version, we leave it alone, and in the rare case when we never see the long version as an expansion of itself, we allow only the short version. We do not use a packed tree structure, because we make far fewer sentences. Additionally, as we are traversing the list of rules to compress the sentences, we keep the list capped at the 100 compressions with the highest $P_{expand}(l \mid s)$. We eventually truncate the list to the best 25, still based upon $P_{expand}(l \mid s)$.

2.3 Special Rules

One difficulty in the use of training data is that so many compressions cannot be modeled by our simple method. The rules it does model, immediate constituent deletion, as in taking out the *ADVP* , of *S* → *ADVP* , *NP VP* ., are certainly common, but many good deletions are more structurally complicated. One particular type of rule, such as *NP(1)* →

NP(2) CC NP(3), where the parent has at least one child with the same label as itself, and the resulting compression is one of the matching children, such as, here, *NP(2)*. There are several hundred rules of this type, and it is very simple to incorporate into our model.

There are other structures that may be common enough to merit adding, but we limit this experiment to the original rules and our new "special rules."

3 Unsupervised Compression

One of the biggest problems with this model of sentence compression is the lack of appropriate training data. Typically, abstracts do not seem to contain short sentences matching long ones elsewhere in a paper, and we would prefer a much larger corpus. Despite this lack of training data, very good results were obtained both by the K&M model and by our variant. We create a way to compress sentences without parallel training data, while sticking as closely to the K&M model as possible.

The source model stays the same, and we still pay a probability cost in the channel model for every subtree deleted. However, the way we determine $P_{expand}(l \mid s)$ changes because we no longer have a parallel text. We create joint rules using only the first section (0.mrg) of the Penn Treebank. We count all probabilistic context free grammar (PCFG) expansions, and then match up similar rules as unsupervised joint events.

We change Equation 2 to calculate $P_{expand}(s \mid l)$ without parallel data. First, let us define svo (shorter version of) to be: r_1 svo r_2 iff the righthand side of r_1 is a subsequence of the righthand side of r_2. Then define

$$P_{expand}(l \mid s) = \frac{count(l)}{\sum_{l' s.t.\ s\ svo\ l'} count(l')} \qquad (3)$$

This is best illustrated by a toy example. Consider a corpus with just 7 rules: 3 instances of *NP* $\rightarrow$ *DT JJ NN* and 4 instances of *NP* $\rightarrow$ *DT NN*.

P(NP $\rightarrow$ *DT JJ NN* $\mid$ *NP* $\rightarrow$ *DT JJ NN)* $= 1$. To determine this, you divide the count of *NP* $\rightarrow$ *DT JJ NN* $= 3$ by all the possible long versions of *NP* $\rightarrow$ *DT JJ NN* $= 3$.

P(NP $\rightarrow$ *DT JJ NN* $\mid$ *NP* $\rightarrow$ *DT NN*) $= 3/7$. The count of *NP* $\rightarrow$ *DT JJ NN* $= 3$, and the possible long versions of *NP* $\rightarrow$ *DT NN* are itself (with count of 3) and *NP* $\rightarrow$ *DT JJ NN* (with count of 4), yielding a sum of 7.

Finally, *P(NP* $\rightarrow$ *DT NN* $\mid$ *NP* $\rightarrow$ *DT NN*) $= 4/7$. The count of *NP* $\rightarrow$ *DT NN* $= 4$, and since the short (*NP* $\rightarrow$ *DT NN*) is the same as above, the count of the possible long versions is again 7.

In this way, we approximate $P_{expand}(l \mid s)$ without parallel data.

Since some of these "training" pairs are likely to be fairly poor compressions, due to the artificiality of the construction, we restrict generation of short sentences to not allow deletion of the head of any subtree. None of the special rules are applied. Other than the above changes, the unsupervised model matches our supervised version. As will be shown, this rule is not constraining enough and allows some poor compressions, but it is remarkable that any sort of compression can be achieved without training data. Later, we will describe additional constraints that help even more.

4 Semi-Supervised Compression

Because the supervised version tends to do quite well, and its main problem is that the model tends to pick longer compressions than a human would, it seems reasonable to incorporate the unsupervised version into our supervised model, in the hope of getting more rules to use. In generating new short sentences, if we have compression probabilities in the supervised version, we use those, including the special rules. The only time we use an unsupervised compression probability is when there is no supervised version of the unsupervised rule.

5 Additional Constraints

Even with the unsupervised constraint from section 3, the fact that we have artificially created our joint rules gives us some fairly ungrammatical compressions. Adding extra constraints improves our unsupervised compressions, and gives us better performance on the supervised version as well. We use a program to label syntactic arguments with the roles they are playing (Blaheta and Charniak, 2000), and the rules for complement/adjunct distinction given by (Collins, 1997) to never allow deletion of the complement. Since many nodes that should not be deleted are not labeled with their syntactic role, we add another constraint that disallows deletion of NPs.

6 Evaluation

As with Knight and Marcu's (2000) original work, we use the same 32 sentence pairs as our Test Corpus, leaving us with 1035 training pairs. After adjusting the supervised weighting parameter, we fold the development set back into the training data.

We presented four judges with nine compressed versions of each of the 32 long sentences: A human-generated short version, the K&M version, our first supervised version, our supervised version with our special rules, our supervised version with special rules and additional constraints, our unsupervised version, our supervised version with additional constraints, our semi-supervised version, and our semi-supervised version with additional constraints. The judges were asked to rate the sentences in two ways: the grammaticality of the short sentences on a scale from 1 to 5, and the importance of the short sentence, or how well the compressed version retained the important words from the original, also on a scale from 1 to 5. The short sentences were randomly shuffled across test cases.

The results in Table 1 show compression rates, as well as average grammar and importance scores across judges.

There are two main ideas to take away from these results. First, we can get good compressions without paired training data. Second, we achieved a good boost by adding our additional constraints in two of the three versions.

Note that importance is a somewhat arbitrary distinction, since according to our judges, *all* of the computer-generated versions do as well in importance as the human-generated versions.

6.1 Examples of Results

In Figure 1, we give four examples of most compression techniques in order to show the range of performance that each technique spans. In the first two examples, we give only the versions with constraints, because there is little or no difference between the versions with and without constraints.

Example 1 shows the additional compression obtained by using our special rules. Figure 2 shows the parse trees of the original pair of short and long versions. The relevant expansion is $NP \rightarrow NP1$, PP in the long version and simply $NP1$ in the short version. The supervised version that includes the special rules learned this particular common special joint rule from the training data and could apply it to the example case. This supervised version compresses better than either version of the supervised noisy-channel model that lacks these rules. The unsupervised version does not compress at all, whereas the semi-supervised version is identical with the better supervised version.

Example 2 shows how unsupervised and semi-supervised techniques can be used to improve compression. Although the final length of the sentences is roughly the same, the unsupervised and semi-supervised versions are able to take the action of deleting the parenthetical. Deleting parentheses was never seen in the training data, so it would be extremely unlikely to occur in this case. The unsupervised version, on the other hand, sees both $PRN \rightarrow lrb\ NP\ rrb$ and $PRN \rightarrow NP$ in its training data, and the semi-supervised version capitalizes on this particular unsupervised rule.

Example 3 shows an instance of our initial supervised versions performing far worse than the K&M model. The reason is that currently our supervised model only generates compressions that it has seen before, unlike the K&M model, which generates all possible compressions. $S \rightarrow S\ ,\ NP\ VP\ .$ never occurs in the training data, and so a good compression does not exist. The unsupervised and semi-supervised versions do better in this case, and the supervised version with the added constraints does even better.

Example 4 gives an example of the K&M model being outperformed by all of our other models.

7 Problems with Noisy Channel Models of Sentence Compression

To this point our presentation has been rather normal; we draw inspiration from a previous paper, and work at improving on it in various ways. We now deviate from the usual by claiming that while the K&M model works very well, there is a technical problem with formulating the task in this way.

We start by making our noisy channel notation a

original:	Many debugging features, including user-defined break points and variable-watching and message-watching windows, have been added.
human:	Many debugging features have been added.
K&M:	Many debugging features, including user-defined points and variable-watching and message-watching windows, have been added.
supervised:	Many features, including user-defined break points and variable-watching and windows, have been added.
super (+ extra rules, constraints):	Many debugging features have been added.
unsuper (+ constraints):	Many debugging features, including user-defined break points and variable-watching and message-watching windows, have been added.
semi-supervised (+ constraints):	Many debugging features have been added.
original:	Also, Trackstar supports only the critical path method (CPM) of project scheduling.
human:	Trackstar supports the critical path method of project scheduling.
K&M:	Trackstar supports only the critical path method (CPM) of scheduling.
supervised:	Trackstar supports only the critical path method (CPM) of scheduling.
super (+ extra rules, constraints):	Trackstar supports only the critical path method (CPM) of scheduling.
unsuper (+ constraints):	Trackstar supports only the critical path method of project scheduling.
semi-supervised (+ constraints):	Trackstar supports only the critical path method of project scheduling.
original:	The faster transfer rate is made possible by an MTI-proprietary data buffering algorithm that off-loads lock-manager functions from the Q-bus host, Raimondi said.
human:	The algorithm off-loads lock-manager functions from the Q-bus host.
K&M:	The faster rate is made possible by a MTI-proprietary data buffering algorithm that off-loads lock-manager functions from the Q-bus host, Raimondi said.
supervised:	Raimondi said.
super (+ extra rules):	Raimondi said.
super (+ extra rules, constraints):	The faster transfer rate is made possible by an MTI-proprietary data buffering algorithm, Raimondi said.
unsuper (+ constraints):	The faster transfer rate is made possible, Raimondi said.
semi-supervised (+ constraints):	The faster transfer rate is made possible, Raimondi said.
original:	The SAS screen is divided into three sections: one for writing programs, one for the system's response as it executes the program, and a third for output tables and charts.
human:	The SAS screen is divided into three sections.
K&M:	The screen is divided into one
super (+ extra rules):	SAS screen is divided into three sections: one for writing programs, and a third for output tables and charts.
super (+ extra rules, constraints):	The SAS screen is divided into three sections.
unsupervised:	The screen is divided into sections: one for writing programs, one for the system's response as it executes program, and third for output tables and charts.
unsupervised (+ constraints):	Screen is divided into three sections: one for writing programs, one for the system's response as it executes program, and a third for output tables and charts.
semi-supervised:	The SAS screen is divided into three sections: one for writing programs, one for the system's response as it executes the program, and a third for output tables and charts.
semi-super (+ constraints):	The screen is divided into three sections: one for writing programs, one for the system's response as it executes the program, and a third for output tables and charts.

Figure 1: Compression Examples

	compression rate	grammar	importance
humans	53.33%	4.96	3.73
K&M	70.37%	4.57	3.85
supervised	79.85%	4.64	3.97
supervised with extra rules	67.41%	4.57	3.66
supervised with extra rules and constraints	68.44%	4.77	3.76
unsupervised	79.11%	4.38	3.93
unsupervised with constraints	77.93%	4.51	3.88
semi-supervised	81.19%	4.79	4.18
semi-supervised with constraints	79.56%	4.75	4.16

Table 1: Experimental Results

short:	(S (NP (JJ Many) (JJ debugging) (NNS features)) (VP (VBP have) (VP (VBN been) (VP (VBN added))))(. .))
long:	(S (NP (NP (JJ Many) (JJ debugging) (NNS features))(, ,) (PP (VBG including) (NP (NP (JJ user-defined)(NN break)(NNS points) (CC and)(NN variable-watching)) (CC and)(NP (JJ message-watching) (NNS windows))))(, ,)) (VP (VBP have) (VP (VBN been) (VP (VBN added))))(. .))

Figure 2: Joint Trees for special rules

bit more explicit:

$$\arg max_s p(s, L = s \mid l, L = l) = \tag{4}$$
$$\arg max_s p(s, L = s) p(l, L = l \mid s, L = s)$$

Here we have introduced explicit conditioning events $L = l$ and $L = s$ to state that that the sentence in question is either the long version or the short version. We do this because in order to get the equation that K&M (and ourselves) start with, it is necessary to assume the following

$$p(s, L = s) = p(s) \tag{5}$$
$$p(l, L = l \mid s, L = s) = p(l \mid s) \tag{6}$$

This means we assume that the probability of, say, s as a short (compressed) sentence is simply its probability as a sentence. This will be, in general, false. One would hope that real compressed sentences are more probable as a member of the set of compressed sentences than they are as simply a member of all English sentences. However, neither K&M, nor we, have a large enough body of compressed and original sentences from which to create useful language models, so we both make this simplifying assumption. At this point it seems like a reasonable choice to make. In fact, it compromises the entire enterprise. To see this, however, we must descend into more details.

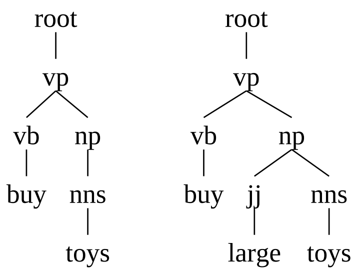


Figure 3: A compression example — trees A and B respectively

Let us consider a simplified version of a K&M example, but as reinterpreted for our model: how the noisy channel model assigns a probability of the compressed tree (A) in Figure 3 given the original tree B.

We compute the probabilities $p(A)$ and $p(B \mid A)$ as follows (Figure 4): We have divided the probabilities up according to whether they are contributed by the source or channel models. Those from the source

$p(A)$	$p(B \mid A)$
$p(\text{s} \rightarrow \text{vp} \mid \text{H}(\text{s}))$	$p(\text{s} \rightarrow \text{vp} \mid \text{s} \rightarrow \text{vp})$
$p(\text{vp} \rightarrow \text{vb np} \mid \text{H}(\text{vp}))$	$p(\text{vp} \rightarrow \text{vb np} \mid \text{vp} \rightarrow \text{vb np})$
$p(\text{np} \rightarrow \text{nns} \mid \text{H}(\text{np}))$	$p(\text{np} \rightarrow \text{jj nns} \mid \text{np} \rightarrow \text{nns})$
$p(\text{vb} \rightarrow \text{buy} \mid \text{H}(\text{vb}))$	$p(\text{vb} \rightarrow \text{buy} \mid \text{vb} \rightarrow \text{buy})$
$p(\text{nns} \rightarrow \text{toys} \mid \text{H}(\text{nns}))$	$p(\text{nns} \rightarrow \text{toys} \mid \text{nns} \rightarrow \text{toys})$
	$p(\text{jj} \rightarrow \text{large} \mid \text{H}(\text{jj}))$

Figure 4: Source and channel probabilities for compressing B into A

$p(B)$	$p(B \mid B)$
$p(\text{s} \rightarrow \text{vp} \mid \text{H}(\text{s}))$	$p(\text{s} \rightarrow \text{vp} \mid \text{s} \rightarrow \text{vp})$
$p(\text{vp} \rightarrow \text{vb np} \mid \text{H}(\text{vp}))$	$p(\text{vp} \rightarrow \text{vb np} \mid \text{vp} \rightarrow \text{vb np})$
$p(\text{np} \rightarrow \text{jj nns} \mid \text{H}(\text{np}))$	$p(\text{np} \rightarrow \text{jj nns} \mid \text{np} \rightarrow \text{jj nns})$
$p(\text{vb} \rightarrow \text{buy} \mid \text{H}(\text{vb}))$	$p(\text{vb} \rightarrow \text{buy} \mid \text{vb} \rightarrow \text{buy})$
$p(\text{nns} \rightarrow \text{toys} \mid \text{H}(\text{nns}))$	$p(\text{nns} \rightarrow \text{toys} \mid \text{nns} \rightarrow \text{toys})$
$p(\text{jj} \rightarrow \text{large} \mid \text{H}(\text{jj}))$	$p(\text{jj} \rightarrow \text{large} \mid \text{jj} \rightarrow \text{large})$

Figure 5: Source and channel probabilities for leaving B as B

model are conditioned on, e.g. $\text{H}(\text{np})$ the history in terms of the tree structure around the noun-phrase. In a pure PCFG this would only include the label of the node. In our language model it includes much more, such as parent and grandparent heads.

Again, following K&M, contrast this with the probabilities assigned when the compressed tree is identical to the original (Figure 5).

Expressed like this it is somewhat daunting, but notice that if all we want is to see which probability is higher (the compressed being the same as the original or truly compressed) then most of these terms cancel, and we get the rule, prefer the truly compressed if and only if the following ratio is greater than one.

$$\frac{p(\text{np} \rightarrow \text{nns} \mid \text{H}(\text{np}))}{p(\text{np} \rightarrow \text{jj nns} \mid \text{H}(\text{np}))} \frac{p(\text{np} \rightarrow \text{jj nns} \mid \text{np} \rightarrow \text{nns})}{p(\text{np} \rightarrow \text{jj nns} \mid \text{np} \rightarrow \text{jj nns})} \frac{1}{p(\text{jj} \rightarrow \text{large} \mid \text{jj} \rightarrow \text{large})} \tag{7}$$

In the numerator are the unmatched probabilities that go into the compressed sentence noisy channel probability, and in the denominator are those for when the sentence does not undergo any change. We can make this even simpler by noting that because tree-bank pre-terminals can only expand into words $p(\text{jj} \rightarrow \text{large} \mid \text{jj} \rightarrow \text{large}) = 1$. Thus the last fraction in Equation 7 is equal to one and can be ignored.

For a compression to occur, it needs to be less desirable to add an adjective in the channel model than in the source model. In fact, the opposite occurs. The likelihood of almost any constituent deletion is far lower than the probability of the constituents all being left in. This seems surprising, considering that the model we are using has had some success, but it makes intuitive sense. There are far fewer compression alignments than total alignments: identical parts of sentences are almost sure to align. So the most probable short sentence should be very barely compressed. Thus we add a weighting factor to compress our supervised version further.

K&M also, in effect, weight shorter sentences more strongly than longer ones based upon their language model. In their papers on sentence compression, they give an example similar to our "buy large toys" example. The equation they get for the channel probabilities in their example is similar to the channel probabilities we give in Figures 3 and 4. However their source probabilities are different. K&M did not have a true syntax-based language model to use as we have. Thus they divided the language model into two parts. Part one assigns probabilities to the grammar rules using a probabilistic context-free grammar, while part two assigns probabilities to the words using a bi-gram model. As they acknowledge in (Knight and Marcu, 2002), the word bigram probabilities are also included in the PCFG probabilities. So in their versions of Figures 3 and 4 they have *both* $p(toys \mid nns)$ (from the PCFG) and $p(toys \mid buy)$ for the bigram probability. In this model, the probabilities do not sum to one, because they pay the probabilistic price for guessing the word "toys" twice, based upon two different conditioning events. Based upon this language model, they prefer shorter sentences.

To reiterate this section's argument: A noisy channel model is *not* by itself an appropriate model for sentence compression. In fact, the most likely short sentence will, in general, be the same length as the long sentence. We achieve compression by weighting to give shorter sentences more likelihood. In fact, what is really required is some model that takes "utility" into account, using a utility model in which shorter sentences are more useful. Our term giving preference to shorter sentences can be thought of as a crude approximation to such a utility. However, this is clearly an area for future research.

8 Conclusion

We have created a supervised version of the noisy-channel model with some improvements over the K&M model. In particular, we learned that adding an additional rule type improved compression, and that enforcing some deletion constraints improves grammaticality. We also show that it is possible to perform an unsupervised version of the compression task, which performs remarkably well. Our semi-supervised version, which we hoped would have good compression rates and grammaticality, had good grammaticality but lower compression than desired.

We would like to come up with a better utility function than a simple weighting parameter for our supervised version. The unsupervised version probably can also be further improved. We achieved much success using syntactic labels to constrain compressions, and there are surely other constraints that can be added.

However, more training data is always the easiest cure to statistical problems. If we can find much larger quantities of training data we could allow for much richer rule paradigms that relate compressed to original sentences. One example of a rule we would like to automatically discover would allow us to compress *all of our design goals* or

```
(NP (NP (DT all))
 (PP (IN of)
  (NP (PRP$ our) (NN design) (NNS goals))))}
```

to *all design goals* or

```
(NP (DT all) (NN design) (NNS goals))
```

In the limit such rules blur the distinction between compression and paraphrase.

9 Acknowledgements

This work was supported by NSF grant IIS-0112435. We would like to thank Kevin Knight and Daniel Marcu for their clarification and test sentences, and Mark Johnson for his comments.

References

Roxana Angheluta, Rudradeb Mitra, Xiuli Jing, and Francine-Marie Moens. 2004. K.U.Leuven summarization system at DUC 2004. In *Document Understanding Conference*.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *The Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 234–240.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. The Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, San Francisco. Morgan Kaufmann.

Gregory Grefenstette. 1998. Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind. In *Working Notes of the AAAI Spring Symposium on Intelligent Text Summarization*, pages 111–118.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: sentence compression. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 703–71.

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

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computationl Linguistics*.

Inderjeet Mani, Barbara Gates, and Eric Bloedorn. 1999. Improving summaries by revising them. In *The Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*. The Association for Computational Linguistics.

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.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. BBN/UMD at DUC 2004: Topiary. In *Document Understanding Conference*.

Digesting Virtual "Geek" Culture: The Summarization of Technical Internet Relay Chats

Liang Zhou and Eduard Hovy
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
{liangz, hovy} @isi.edu

Abstract

This paper describes a summarization system for technical chats and emails on the Linux kernel. To reflect the complexity and sophistication of the discussions, they are clustered according to subtopic structure on the sub-message level, and immediate responding pairs are identified through machine learning methods. A resulting summary consists of one or more mini-summaries, each on a subtopic from the discussion.

1 Introduction

The availability of many chat forums reflects the formation of globally dispersed virtual communities. From them we select the very active and growing movement of Open Source Software (OSS) development. Working together in a virtual community in non-collocated environments, OSS developers communicate and collaborate using a wide range of web-based tools including Internet Relay Chat (IRC), electronic mailing lists, and more (Elliott and Scacchi, 2004). In contrast to conventional instant message chats, IRCs convey engaging and focused discussions on collaborative software development. Even though all OSS participants are technically savvy individually, summaries of IRC content are necessary within a virtual organization both as a resource and an organizational memory of activities (Ackerman and Halverson, 2000). They are regularly produced manually by volunteers. These summaries can be used for analyzing the impact of virtual social interactions and virtual organizational culture on software/product development.

The emergence of email thread discussions and chat logs as a major information source has prompted increased interest in thread summarization within the Natural Language Processing (NLP) community. One might assume a smooth transition from text-based summarization to email and chat-based summarizations. However, chat falls in the genre of correspondence, which requires dialogue and conversation analysis. This property makes summarization in this area even more difficult than traditional summarization. In particular, topic "drift" occurs more radically than in written genres, and interpersonal and pragmatic content appears more frequently. Questions about the content and overall organization of the summary must be addressed in a more thorough way for chat and other dialogue summarization systems.

In this paper we present a new system that clusters sub-message segments from correspondences according to topic, identifies the sub-message segment containing the leading issue within the topic, finds immediate responses from other participants, and consequently produces a summary for the entire IRC. Other constructions are possible. One of the two baseline systems described in this paper uses the timeline and dialogue structure to select summary content, and is quite effective. We use the term *chat* loosely in this paper. Input IRCs for our system is a mixture of chats and

Proceedings of the 43rd Annual Meeting of the ACL, pages 298–305,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

emails that are indistinguishable in format observed from the downloaded corpus (Section 3).

In the following sections, we summarize previous work, describe the email/chat data, intra-message clustering and summary extraction process, and discuss the results and future work.

2 Previous and Related Work

There are at least two ways of organizing dialogue summaries: by dialogue structure and by topic.

Newman and Blitzer (2002) describe methods for summarizing archived newsgroup conversations by clustering messages into subtopic groups and extracting top-ranked sentences per subtopic group based on the intrinsic scores of position in the cluster and lexical centrality. Due to the technical nature of our working corpus, we had to handle intra-message topic shifts, in which the author of a message raises or responds to multiple issues in the same message. This requires that our clustering component be not message-based but sub-message-based.

Lam et al. (2002) employ an existing summarizer for single documents using preprocessed email messages and context information from previous emails in the thread.

Rambow et al. (2004) show that sentence extraction techniques are applicable to summarizing email threads, but only with added email-specific features. Wan and McKeown (2004) introduce a system that creates overview summaries for ongoing decision-making email exchanges by first detecting the issue being discussed and then extracting the response to the issue. Both systems use a corpus that, on average, contains 190 words and 3.25 messages per thread, much shorter than the ones in our collection.

Galley et al. (2004) describe a system that identifies agreement and disagreement occurring in human-to-human multi-party conversations. They utilize an important concept from conversational analysis, adjacent pairs (AP), which consists of initiating and responding utterances from different speakers. Identifying APs is also required by our research to find correspondences from different chat participants.

In automatic summarization of spoken dialogues, Zechner (2001) presents an approach to obtain extractive summaries for multi-party dialogues in unrestricted domains by addressing intrinsic issues specific to speech transcripts. Automatic question detection is also deemed important in this work. A decision-tree classifier was trained on question-triggering words to detect questions among speech acts (sentences). A search heuristic procedure then finds the corresponding answers. Ries (2001) shows how to use keyword repetition, speaker initiative and speaking style to achieve topical segmentation of spontaneous dialogues.

3 Technical Internet Relay Chats

GNUe, a meta-project of the GNU project[1]–one of the most famous free/open source software projects–is the case study used in (Elliott and Scacchi, 2004) in support of the claim that, even in virtual organizations, there is still the need for successful conflict management in order to maintain order and stability.

The GNUe IRC archive is uniquely suited for our experimental purpose because each IRC chat log has a companion summary digest written by project participants as part of their contribution to the community. This manual summary constitutes gold-standard data for evaluation.

3.1 Kernel Traffic[2]

Kernel Traffic is a collection of summary digests of discussions on GNUe development. Each digest summarizes IRC logs and/or email messages (later referred to as chat logs) for a period of up to two weeks. A nice feature is that direct quotes and hyperlinks are part of the summary. Each digest is an extractive overview of facts, plus the author's dramatic and humorous interpretations.

3.2 Corpus Download

The complete Linux Kernel Archive (LKA) consists of two separate downloads. The Kernel Traffic (summary digests) are in XML format and were downloaded by crawling the Kernel Traffic site. The Linux Kernel Archives (individual IRC chat logs) are downloaded from the archive site. We matched the summaries with their respective chat logs based on subject line and publication dates.

3.3 Observation on Chat Logs

[1] http://www.gnu.org
[2] http://kt.hoser.ca/kernel-traffic/index.html

Upon initial examination of the chat logs, we found that many conventional assumptions about chats in general do not apply. For example, in most instant-message chats, each exchange usually consists of a small number of words in several sentences. Due to the technical nature of GNUe, half of the chat logs contain in-depth discussions with lengthy messages. One message might ask and answer several questions, discuss many topics in detail, and make further comments. This property, which we call *subtopic structure*, is an important difference from informal chat/interpersonal banter. Figure 1 shows the subtopic structure and relation of the first 4 messages from a chat log, produced manually. Each message is represented horizontally; the vertical arrows show where participants responded to each other. Visual inspection reveals in this example there are three distinctive clusters (a more complex cluster and two smaller satellite clusters) of discussions between participants at sub-message level.

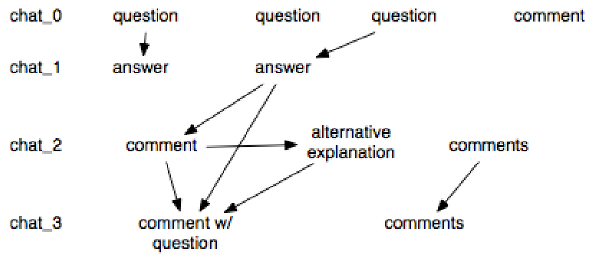


Figure 1. An example of chat subtopic structure and relation between correspondences.

3.4 Observation on Summary Digests

To measure the goodness of system-produced summaries, *gold standards* are used as references. Human-written summaries usually make up the gold standards. The Kernel Traffic (summary digests) are written by Linux experts who actively contribute to the production and discussion of the open source projects. However, participant-produced digests cannot be used as reference summaries verbatim. Due to the complex structure of the dialogue, the summary itself exhibits some discourse structure, necessitating such reader guidance phrases such as "for the … question," "on the … subject," "regarding …," "later in the same thread," etc., to direct and refocus the reader's attention. Therefore, further manual editing and partitioning is needed to transform a multi-topic digest into several smaller subtopic-based *gold-standard* reference summaries (see Section 6.1 for the transformation).

4 Fine-grained Clustering

To model the subtopic structure of each chat message, we apply clustering at the sub-message level.

4.1 Message Segmentation

First, we look at each message and assume that each participant responds to an ongoing discussion by stating his/her opinion on several topics or issues that have been discussed in the current chat log, but not necessarily in the order they were discussed. Thus, topic shifts can occur sequentially within a message. Messages are partitioned into multi-paragraph segments using TextTiling, which reportedly has an overall precision of 83% and recall of 78% (Hearst, 1994).

4.2 Clustering

After distinguishing a set of message segments, we cluster them. When choosing an appropriate clustering method, because the number of subtopics under discussion is unknown, we cannot make an assumption about the total number of resulting clusters. Thus, nonhierarchical partitioning methods cannot be used, and we must use a hierarchical method. These methods can be either *agglomerative*, which begin with an unclustered data set and perform $N - 1$ pairwise joins, or *divisive*, which add all objects to a single cluster, and then perform $N - 1$ divisions to create a hierarchy of smaller clusters, where N is the total number of items to be clustered (Frakes and Baeza-Yates, 1992).

Ward's Method

Hierarchical agglomerative clustering methods are commonly used and we employ Ward's method (Ward and Hook, 1963), in which the text segment pair merged at each stage is the one that minimizes the increase in total within-cluster variance.

Each cluster is represented by an L-dimensional vector $(x_{i1}, x_{i2}, \ldots, x_{iL})$ where each x_{ik} is the word's $tf \bullet idf$ score. If m_i is the number of objects in the cluster, the squared Euclidean distance between two segments i and j is:

$$d_{ij}^2 = \sum_{K=1}^{L} (x_{ik} - x_{jk})^2$$

When two segments are joined, the increase in variance I_{ij} is expressed as:

$$I_{ij} = \frac{m_i m_j}{m_i + m_j} d_{ij}^2$$

Number of Clusters

The process of joining clusters continues until the combination of any two clusters would destabilize the entire array of currently existing clusters produced from previous stages. At each stage, the two clusters x_{ik} and x_{jk} are chosen whose combination would cause the minimum increase in variance I_{ij}, expressed as a percentage of the variance change from the last round. If this percentage reaches a preset threshold, it means that the nearest two clusters are much further from each other compared to the previous round; therefore, joining of the two represents a destabilizing change, and should not take place.

Sub-message segments from resulting clusters are arranged according to the sequence the original messages were posted and the resulting subtopic structures are similar to the one shown in Figure 1.

5 Summary Extraction

Having obtained clusters of message segments focused on subtopics, we adopt the typical summarization paradigm to extract informative sentences and segments from each cluster to produce subtopic-based summaries. If a chat log has n clusters, then the corresponding summary will contain n mini-summaries.

All message segments in a cluster are related to the central topic, but to various degrees. Some are answers to questions asked previously, plus further elaborative explanations; some make suggestions and give advice where they are requested, etc. From careful analysis of the LKA data, we can safely assume that for this type of conversational interaction, the goal of the participants is to seek help or advice and advance their current knowledge on various technical subjects. This kind of interaction can be modeled as one problem-initiating segment and one or more corresponding problem-solving segments. We envisage that identifying corresponding message segment pairs will produce adequate summaries. This analysis follows the structural organization of summaries from Kernel Traffic. Other types of discussions, at least in part, require different discourse/summary organization.

These corresponding pairs are formally introduced below, and the methods we experimented with for identifying them are described.

5.1 Adjacent Response Pairs

An important conversational analysis concept, adjacent pairs (AP), is applied in our system to identify initiating and responding correspondences from different participants in one chat log. Adjacent pairs are considered fundamental units of conversational organization (Schegloff and Sacks, 1973). An adjacent pair is said to consist of two parts that are ordered, adjacent, and produced by different speakers (Galley et al., 2004). In our email/chat (LKA) corpus a physically adjacent message, following the timeline, may not directly respond to its immediate predecessor. Discussion participants read the current live thread and decide what he/she would like to correspond to, not necessarily in a serial fashion. With the added complication of subtopic structure (see Figure 1) the definition of adjacency is further violated. Due to its problematic nature, a relaxation on the adjacency requirement is used in extensive research in conversational analysis (Levinson, 1983). This relaxed requirement is adopted in our research.

Information produced by adjacent correspondences can be used to produce the subtopic-based summary of the chat log. As described in Section 4, each chat log is partitioned, at sub-message level, into several subtopic clusters. We take the message segment that appears first chronologically in the cluster as the topic-initiating segment in an adjacent pair. Given the initiating segment, we need to identify one or more segments from the same cluster that are the most direct and relevant responses. This process can be viewed equivalently as the informative sentence extraction process in conventional text-based summarization.

5.2 AP Corpus and Baseline

We manually tagged 100 chat logs for adjacent pairs. There are, on average, 11 messages per chat log and 3 segments per message (This is considerably larger than threads used in previous research). Each chat log has been clustered into one or more bags of message segments. The message segment that appears earliest in time in a cluster was marked as the initiating segment. The annotators were provided with this segment and one other segment at a time, and were asked to decide whether the current message segment is a direct answer to the question asked, the suggestion that was requested, etc. in the initiating segment. There are 1521 adjacent response pairs; 1000 were used for training and 521 for testing.

Our baseline system selects the message segment (from a different author) immediately following the initiating segment. It is quite effective, with an accuracy of 64.67%. This is reasonable because not all adjacent responses are interrupted by messages responding to different earlier initiating messages.

In the following sections, we describe two machine learning methods that were used to identify the second element in an adjacent response pair and the features used for training. We view the problem as a binary classification problem, distinguishing less relevant responses from direct responses. Our approach is to assign a candidate message segment *c* an appropriate response class *r*.

5.3 Features

Structural and durational features have been demonstrated to improve performance significantly in conversational text analysis tasks. Using them, Galley et al. (2004) report an 8% increase in speaker identification. Zechner (2001) reports excellent results (F > .94) for inter-turn sentence boundary detection when recording the length of pause between utterances. In our corpus, durational information is nonexistent because chats and emails were mixed and no exact time recordings beside dates were reported. So we rely solely on structural and lexical features.

For structural features, we count the number of messages between the initiating message segment and the responding message segment. Lexical features are listed in Table 1. The tech words are the words that are uncommon in conventional literature and unique to Linux discussions.

5.4 Maximum Entropy

Maximum entropy has been proven to be an effective method in various natural language processing applications (Berger et al., 1996). For training and testing, we used YASMET[3]. To est imate *P(r | c)* in the exponential form, we have:

$$P_\lambda(r \mid c) = \frac{1}{Z_\lambda(c)} \exp\left(\sum_i \lambda_{i,r} f_{i,r}(c,r)\right)$$

where $Z_\lambda(c)$ is a normalizing constant and the feature function for feature f_i and response class r is defined as:

$$f_{i,r}(c,r') = \begin{cases} 1, & \text{if } f_i > 0 \text{ and } r' = r \\ 0, & \text{otherwise} \end{cases} .$$

$\lambda_{i,r}$ is the feature-weight parameter for feature f_i and response class r. Then, to determine the best class r for the candidate message segment c, we have:

$$r^* = \arg\max_r P(r \mid c) \quad .$$

- number of overlapping words
- number of overlapping content words
- ratio of overlapping words
- ratio of overlapping content words
- number of overlapping tech words

Table 1. Lexical features.

5.5 Support Vector Machine

Support vector machines (SVMs) have been shown to outperform other existing methods (naïve Bayes, k-NN, and decision trees) in text categorization (Joachims, 1998). Their advantages are robustness and the elimination of the need for feature selection and parameter tuning. SVMs find the hyperplane that separates the positive and negative training examples with maximum margin. Finding this hyperplane can be translated into an optimization problem of finding a set of coefficients α_i^* of the weight vector $\vec{w}$ for document d_i of class $y_i \in \{+1, -1\}$:

$$\vec{w} = \sum_i \alpha_i^* y_i \vec{d}_i, \quad \alpha_i > 0 \quad .$$

Testing data are classified depending on the side of the hyperplane they fall on. We used the LIBSVM[4] package for training and testing.

Feature sets	baseline	MaxEnt	SVM
	64.67%		
Structural		61.22%	71.79%
Lexical		62.24%	72.22%
Structural + Lexical		72.61%	72.79%

Table 2. Accuracy on identifying APs.

[3] http://www.fjoch.com/YASMET.html
[4] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

5.6 Results

Entries in Table 2 show the accuracies achieved using machine learning models and feature sets.

5.7 Summary Generation

After responding message segments are identified, we couple them with their respective initiating segment to form a mini-summary based on their subtopic. Each initializing segment has zero or more responding segments. We also observed *zero response* in human-written summaries where participants initiated some question or concern, but others failed to follow up on the discussion. The AP process is repeated for each cluster created previously. One or more subtopic-based mini-summaries make up one final summary for each chat log. Figure 2 shows an example. For longer chat logs, the length of the final summary is arbitrarily averaged at 35% of the original.

Subtopic 1:
Benjamin Reed: I wrote a wireless ethernet driver a while ago... Are driver writers recommended to use that over extending **/proc** or is it deprecated?
Linus Torvalds: **Syscyl** is deprecated. It's useful in one way only ...

Subtopic 2:
Benjamin Reed: I am a bit uncomfortable ... wondering for a while if there are **guidelines** on …
Linus Torvalds: The thing to do is to create ...

Subtopic 3:
Marcin Dalecki: Are you just blind to the never-ending **format/ compatibility/ … problems** the whole idea behind /proc induces inherently?

Figure 2. A system-produced summary.

6 Summary Evaluation

To evaluate the goodness of the system-produced summaries, a set of reference summaries is used for comparison. In this section, we describe the manual procedure used to produce the reference summaries, and the performances of our system and two baseline systems.

6.1 Reference Summaries

Kernel Traffic digests are participant-written summaries of the chat logs. Each digest mixes the summary writer's own narrative comments with direct quotes (citing the authors) from the chat log. As observed in Section 3.4, subtopics are intermingled in each digest. Authors use key phrases to link the contents of each subtopic throughout texts. In Figure 3, we show an example of such a digest. Discussion participants' names are in italics and subtopics are in bold. In this example, the conversation was started by Benjamin Reed with two questions: 1) asking for conventions for writing /proc drivers, and 2) asking about the status of sysctl. The summary writer indicated that Linus Torvalds replied to both questions and used the phrase "for the … question, he added…" to highlight the answer to the second question. As the di-

Benjamin Reed wrote a wireless Ethernet **driver that used /proc** as its interface. But he was a little uncomfortable … asked if there were any conventions he should follow. He added, "and finally, what's up with **sysctl**? …"
Linus Torvalds replied with: "the thing to do is to create a …[program code]. The **/proc/drivers/** directory is already there, so you'd basically do something like … [program code]." For the **sysctl** question, he added "sysctl is deprecated. ..."
Marcin Dalecki flamed Linus: "Are you just blind to the never-ending format/compatibility/… problems the whole idea behind **/proc** induces inherently? …[example]"

Figure 3. An original Kernel Traffic digest.

Mini 1:
Benjamin Reed wrote a wireless Ethernet **driver that used /proc** as its interface. But he was a little uncomfortable … and asked if there were any conventions he should follow.
Linus Torvalds replied with: the thing to do is to create a …[program code]. The **/proc/drivers/** directory is already there, so you'd basically do something like … [program code].
Marcin Dalecki flamed Linus: Are you just blind to the never-ending format/ compatibility/ … problems the whole idea behind **/proc** induces inherently? …[example]

Mini 2:
Benjamin Reed: and finally, what's up with **sysctl**? ...
Linus Torvalds replied: sysctl is deprecated. ...

Figure 4. A reference summary reproduced from a summary digest.

gest goes on, Marcin Dalecki only responded to the first question with his excited commentary.

Since our system-produced summaries are subtopic-based and partitioned accordingly, if we use unprocessed Kernel Traffic as references, the comparison would be rather complicated and would increase the level of inconsistency in future assessments. We manually reorganized each summary digest into one or more mini-summaries by subtopic (see Figure 4.) Examples (usually kernel stats) and programs are reduced to "[example]" and "[program code]." Quotes (originally in separate messages but merged by the summary writer) that contain multiple topics are segmented and the participant's name is inserted for each segment. We follow clues like "to answer … question" to pair up the main topics and their responses.

6.2 Summarization Results

We evaluated 10 chat logs. On average, each contains approximately 50 multi-paragraph tiles (partitioned by TextTile) and 5 subtopics (clustered by the method from Section 4).

A simple baseline system takes the first sentence from each email in the sequence that they were posted, based on the assumption that people tend to put important information in the beginning of texts (*Position Hypothesis*).

A second baseline system was built based on constructing and analyzing the dialogue structure of each chat log. Participants often quote portions of previously posted messages in their responses. These quotes link most of the messages from a chat log. The message segment that immediately follows the quote is automatically paired with the quote itself and added to the summary and sorted according to the timeline. Segments that are not quoted in later messages are labeled as less relevant and discarded. A resulting baseline summary is an inter-connected structure of segments that quoted and responded to one another. Figure 5 is a shortened summary produced by this baseline for the ongoing example.

[0|0] *Benjamin Reed*: "I wrote an … driver … /proc …"
[0|1] *Benjamin Reed*: "… /proc/ guideline …"
[0|2] *Benjamin Reed*: "… syscyl …"
[1|0] *Linus Torvalds* responds to [0|0, 0|1, 0|2]: "the thing to do is …" "sysctl is deprecated … "

Figure 5. A short example from Baseline 2.

The summary digests from Kernel Traffic mostly consist of direct snippets from original messages, thus making the reference summaries extractive even after rewriting. This makes it possible to conduct an automatic evaluation. A computerized procedure calculates the overlap between reference and system-produced summary units. Since each system-produced summary is a set of mini-summaries based on subtopics, we also compared the subtopics against those appearing in reference summaries (precision = 77.00%, recall = 74.33 %, F = 0.7566).

		Recall	Precision	F-measure
Baseline1		30.79%	16.81%	.2175
Baseline2		63.14%	36.54%	.4629
System	Summary	52.57%	52.14%	.5235
	Topic-summ	52.57%	63.66%	.5758

Table 3. Summary of results.

Table 3 shows the *recall*, *precision*, and *F-measure* from the evaluation. From manual analysis on the results, we notice that the original digest writers often leave large portions of the discussion out and focus on a few topics. We think this is because among the participants, some are Linux veterans and others are novice programmers. Digest writers recognize this difference and reflect it in their writings, whereas our system does not. The entry "*Topic-summ*" in the table shows system-produced summaries being compared only against the topics discussed in the reference summaries.

6.3 Discussion

A recall of 30.79% from the simple baseline reassures us the Position Hypothesis still applies in conversational discussions. The second baseline performs extremely well on recall, 63.14%. It shows that quoted message segments, and thereby derived dialogue structure, are quite indicative of where the important information resides. Systems built on these properties are good summarization systems and hard-to-beat baselines. The system described in this paper (*Summary*) shows an *F-measure* of .5235, an improvement from .4629 of the smart baseline. It gains from a high precision because less relevant message segments are identified and excluded from the adjacent response pairs, leaving mostly topic-oriented segments in summaries. There is a slight improvement when assessing against only those subtopics appeared in the reference summaries (*Topic-summ*). This shows that we only identified clusters on their information content, not on their respective writers' experience and reliability of knowledge.

In the original summary digests, interactions and reactions between participants are sometimes described. Digest writers insert terms like "flamed", "surprised", "felt sorry", "excited", etc. To analyze social and organizational culture in a virtual environment, we need not only information extracts (implemented so far) but also passages that reveal the personal aspect of the communications. We plan to incorporate opinion identification into the current system in the future.

7 Conclusion and Future Work

In this paper we have described a system that performs intra-message topic-based summarization by clustering message segments and classifying topic-initiating and responding pairs. Our approach is an initial step in developing a framework that can eventually reflect the human interactions in virtual environments. In future work, we need to prioritize information according to the perceived knowledgeability of each participant in the discussion, in addition to identifying informative content and recognizing dialogue structure. While the approach to the detection of initiating-responding pairs is quite effective, differentiating important and non-important topic clusters is still unresolved and must be explored.

References

M. S. Ackerman and C. Halverson. 2000. Reexaming organizational memory. *Communications of the ACM*, 43(1), 59–64.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

M. Elliott and W. Scacchi. 2004. Free software development: cooperation and conflict in a virtual organizational culture. S. Koch (ed.), *Free/Open Source Software Development*, IDEA publishing, 2004.

W. B. Frakes and R. Baeza-Yates. 1992. Information retrieval: data structures & algorithms. Prentice Hall.

M. Galley, K. McKeown, J. Hirschberg, and E. Shriberg. 2004. Identifying agreement and disagreement in conversational speech: use of Bayesian networks to model pragmatic dependencies. In the *Proceedings of ACL-04*.

M. A. Hearst. 1994. Multi-paragraph segmentation of expository text. In the *Proceedings of ACL 1994*.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the ECML*, pages 137–142.

D. Lam and S. L. Rohall. 2002. Exploiting e-mail structure to improve summarization. *Technical Paper at IBM Watson Research Center* #20–02.

S. Levinson. 1983. *Pragmatics*. Cambridge University Press.

P. Newman and J. Blitzer. 2002. Summarizing archived discussions: a beginning. In *Proceedings of Intelligent User Interfaces*.

O. Rambow, L. Shrestha, J. Chen and C. Laurdisen. 2004. Summarizing email threads. In *Proceedings of HLT-NAACL 2004: Short Papers*.

K. Ries. 2001. Segmenting conversations by topic, initiative, and style. In *Proceedings of SIGIR Workshop: Information Retrieval Techniques for Speech Applications 2001*: 51–66.

E. A. Schegloff and H. Sacks. 1973. Opening up closings. *Semiotica*, 7-4:289–327.

S. Wan and K. McKeown. 2004. Generating overview summaries of ongoing email thread discussions. In *Proceedings of COLING 2004*.

J. H. Ward Jr. and M. E. Hook. 1963. Application of an hierarchical grouping procedure to a problem of grouping profiles. *Educational and Psychological Measurement*, 23, 69–81.

K. Zechner. 2001. Automatic generation of concise summaries of spoken dialogues in unrestricted domains. In *Proceedings of SIGIR 2001*.

Lexicalization in Crosslinguistic Probabilistic Parsing: The Case of French

Abhishek Arun and **Frank Keller**
School of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW, UK
a.arun@sms.ed.ac.uk, keller@inf.ed.ac.uk

Abstract

This paper presents the first probabilistic parsing results for French, using the recently released French Treebank. We start with an unlexicalized PCFG as a baseline model, which is enriched to the level of Collins' Model 2 by adding lexicalization and subcategorization. The lexicalized sister-head model and a bigram model are also tested, to deal with the flatness of the French Treebank. The bigram model achieves the best performance: 81% constituency F-score and 84% dependency accuracy. All lexicalized models outperform the unlexicalized baseline, consistent with probabilistic parsing results for English, but contrary to results for German, where lexicalization has only a limited effect on parsing performance.

1 Introduction

This paper brings together two strands of research that have recently emerged in the field of probabilistic parsing: crosslinguistic parsing and lexicalized parsing. Interest in parsing models for languages other than English has been growing, starting with work on Czech (Collins et al., 1999) and Chinese (Bikel and Chiang, 2000; Levy and Manning, 2003). Probabilistic parsing for German has also been explored by a range of authors (Dubey and Keller, 2003; Schiehlen, 2004). In general, these authors have found that existing lexicalized parsing models for English (e.g., Collins 1997) do not straightforwardly generalize to new languages; this typically manifests itself in a severe reduction in parsing performance compared to the results for English.

A second recent strand in parsing research has dealt with the role of lexicalization. The conventional wisdom since Magerman (1995) has been that lexicalization substantially improves performance compared to an unlexicalized baseline model (e.g., a probabilistic context-free grammar, PCFG). However, this has been challenged by Klein and Manning (2003), who demonstrate that an unlexicalized model can achieve a performance close to the state of the art for lexicalized models. Furthermore, Bikel (2004) provides evidence that lexical information (in the form of bi-lexical dependencies) only makes a small contribution to the performance of parsing models such as Collins's (1997).

The only previous authors that have directly addressed the role of lexicalization in crosslinguistic parsing are Dubey and Keller (2003). They show that standard lexicalized models fail to outperform an unlexicalized baseline (a vanilla PCFG) on Negra, a German treebank (Skut et al., 1997). They attribute this result to two facts: (a) The Negra annotation assumes very flat trees, which means that Collins-style head-lexicalization fails to pick up the relevant information from non-head nodes. (b) German allows flexible word order, which means that standard parsing models based on context free grammars perform poorly, as they fail to generalize over different positions of the same constituent.

As it stands, Dubey and Keller's (2003) work does not tell us whether treebank flatness or word order flexibility is responsible for their results: for English, the annotation scheme is non-flat, and the word order is non-flexible; lexicalization improves performance. For German, the annotation scheme is flat and the word order is flexible; lexicalization fails to improve performance. The present paper provides the missing piece of evidence by applying probabilistic parsing models to French, a language with non-flexible word order (like English), but with a treebank with a flat annotation scheme (like German). Our results show that French patterns with English: a large increase of parsing performance can be obtained by using a lexicalized model. We conclude that the failure to find a sizable effect of lexicalization in German can be attributed to the word order flexibility of that language, rather than to the flatness of the annotation in the German treebank.

The paper is organized as follows: In Section 2, we give an overview of the French Treebank we use for our experiments. Section 3 discusses its annotation scheme and introduces a set of tree transformations that we apply. Section 4 describes the pars-

Proceedings of the 43rd Annual Meeting of the ACL, pages 306–313,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

```
<NP>
<w lemma="eux" ei="PROmp"
   ee="PRO-3mp" cat="PRO"
   subcat="3mp">eux</w>
</NP>
```

Figure 1: Word-level annotation in the French Treebank: *eux* 'they' (`cat`: POS tag, `subcat`: subcategory, `ei`, `ee`: inflection)

```
<w compound="yes" lemma="d'entre"
   ei="P" ee="P" cat="P">
   <w catint="P">d'</w>
   <w catint="P">entre</w>
</w>
```

Figure 2: Annotation of compounds in the French Treebank: *d'entre* 'between' (`catint`: compound-internal POS tag)

ing models, followed by the results for the unlexicalized baseline model in Section 6 and for a range of lexicalized models in Section 5. Finally, Section 7 provides a crosslinguistic comparison involving data sets of the same size extracted from the French, English, and German treebanks.

2 The French Treebank

2.1 Annotation Scheme

The French Treebank (FTB; Abeillé et al. 2000) consists of 20,648 sentences extracted from the daily newspaper *Le Monde*, covering a variety of authors and domains (economy, literature, politics, etc.).[1] The corpus is formatted in XML and has a rich morphosyntactic tagset that includes part-of-speech tag, 'subcategorization' (e.g., possessive or cardinal), inflection (e.g., masculine singular), and lemma information. Compared to the Penn Treebank (PTB; Marcus et al. 1993), the POS tagset of the French Treebank is smaller (13 tags vs. 36 tags): all punctuation marks are represented as the single PONCT tag, there are no separate tags for modal verbs, *wh*-words, and possessives. Also verbs, adverbs and prepositions are more coarsely defined. On the other hand, a separate clitic tag (CL) for weak pronouns is introduced. An example for the word-level annotation in the FTB is given in Figure 1

The phrasal annotation of the FTB differs from that for the Penn Treebank in several aspects. There is no verb phrase: only the *verbal nucleus* (VN) is annotated. A VN comprises the verb and any clitics, auxiliaries, adverbs, and negation associated with it. This results in a flat syntactic structure, as in (1).

(1) (VN (V sont) (ADV systématiquement) (V arrêtés)) 'are systematically arrested'

The noun phrases (NPs) in the FTB are also flat; a noun is grouped together with any associated determiners and prenominal adjectives, as in example (2). Note that postnominal adjectives, however, are adjoined to the NP in an adjectival phrase (AP).

(2) (NP (D des) (A petits) (N mots) (AP (ADV très) (A gentils))) 'small, very gentle words'

Unlike the PTB, the FTB annotates *coordinated phrases* with the syntactic tag COORD (see the left panel of Figure 3 for an example).

The treatment of *compounds* is also different in the FTB. Compounds in French can comprise words which do not exist otherwise (e.g., *insu* in the compound preposition *à l'insu de* 'unbeknownst to') or can exhibit sequences of tags otherwise ungrammatical (e.g., *à la va vite* 'in a hurry': Prep + Det + finite verb + adverb). To account for these properties, compounds receive a two-level annotation in the FTB: a subordinate level is added for the constituent parts of the compound (both levels use the same POS tagset). An example is given in Figure 2.

Finally, the FTB differs from the PTB in that it does not use any empty categories.

2.2 Data Sets

The version of the FTB made available to us (version 1.4, May 2004) contains numerous errors. Two main classes of inaccuracies were found in the data: (a) The word is present but morphosyntactic tags are missing; 101 such cases exist. (b) The tag information for a word (or a part of a compound) is present but the word (or compound part) itself is missing. There were 16,490 instances of this error in the dataset.

Initially we attempted to correct the errors, but this proved too time consuming, and we often found that the errors cannot be corrected without access to the raw corpus, which we did not have. We therefore decided to remove all sentences with errors, which lead to a reduced dataset of 10,552 sentences.

The remaining data set (222,569 words at an average sentence length of 21.1 words) was split into a training set, a development set (used to test the parsing models and to tune their parameters), and a test set, unseen during development. The training set consisted of the first 8,552 sentences in the corpus, with the following 1000 sentences serving as the development set and the final 1000 sentences forming the test set. All results reported in this paper were obtained on the test set, unless stated otherwise.

[1]The French Treebank was developed at Université Paris 7. A license can be obtained by emailing Anne Abeillé (`abeille@linguist.jussieu.fr`).

3 Tree Transformations

We created a number of different datasets from the FTB, applying various tree transformation to deal with the peculiarities of the FTB annotation scheme. As a first step, the XML formatted FTB data was converted to PTB-style bracketed expressions. Only the POS tag was kept and the rest of the morphological information for each terminal was discarded. For example, the NP in Figure 1 was transformed to:

(3) (NP (PRO eux))

In order to make our results comparable to results from the literature, we also transformed the annotation of punctuation. In the FTB, all punctuations is tagged uniformly as PONCT. We reassigned the POS for punctuation using the PTB tagset, which differentiates between commas, periods, brackets, etc.

Compounds have internal structure in the FTB (see Section 2.1). We created two separate data sets by applying two alternative tree transformation to make FTB compounds more similar to compounds in other annotation schemes. The first was *collapsing the compound* by concatenating the compound parts using an underscore and picking up the `cat` information supplied at the compound level. For example, the compound in Figure 2 results in:

(4) (P d'_entre)

This approach is similar to the treatment of compounds in the German Negra treebank (used by Dubey and Keller 2003), where compounds are not given any internal structure (compounds are mostly spelled without spaces or apostrophes in German).

The second approach is *expanding the compound*. Here, the compound parts are treated as individual words with their own POS (from the `catint` tag), and the suffix `Cmp` is appended the POS of the compound, effectively expanding the tagset.[2] Now Figure 2 yields:

(5) (PCmp (P d') (P entre)).

This approach is similar to the treatment of compounds in the PTB (except hat the PTB does not use a separate tag for the mother category). We found that in the FTB the POS tag of the compound part is sometimes missing (i.e., the value of `catint` is blank). In cases like this, the missing `catint` was substituted with the `cat` tag of the compound. This heuristic produces the correct POS for the subparts of the compound most of the time.

[2]An alternative would be to retain the `cat` tag of the compound. The effect of this decision needs to be investigated in future work.

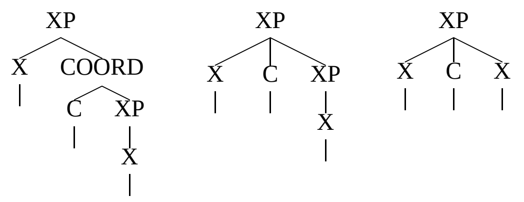


Figure 3: Coordination in the FTB: before (left) and after transformation (middle); coordination in the PTB (right)

As mentioned previously, coordinate structures have their own constituent label COORD in the FTB annotation. Existing parsing models (e.g., the Collins models) have coordination-specific rules, presupposing that coordination is marked up in PTB format. We therefore created additional datasets where a transformation is applied that *raises coordination*. This is illustrated in Figure 3. Note that in the FTB annotation scheme, a coordinating conjunction is always followed by a syntactic category. Hence the resulting tree, though flatter, is still not fully compatible with the PTB treatment of coordination.

4 Probabilistic Parsing Models

4.1 Probabilistic Context-Free Grammars

The aim of this paper is to further explore the crosslinguistic role of lexicalization by applying lexicalized parsing models to the French Treebank parsing accuracy. Following Dubey and Keller (2003), we use a standard unlexicalized PCFG as our baseline. In such a model, each context-free rule $RHS \rightarrow LHS$ is annotated with an expansion probability $P(RHS|LHS)$. The probabilities for all the rules with the same left-hand side have to sum up to one and the probability of a parse tree T is defined as the product of the probabilities of each rule applied in the generation of T.

4.2 Collins' Head-Lexicalized Models

A number of lexicalized models can then be applied to the FTB, comparing their performance to the unlexicalized baseline. We start with Collins' Model 1, which lexicalizes a PCFG by associating a word w and a POS tag t with each non-terminal X in the tree. Thus, a non-terminal is written as $X(x)$ where $x = \langle w,t \rangle$ and X is constituent label. Each rule now has the form:

(1) $P(h) \rightarrow L_n(l_n) \dots L_1(l_1) H(h) R_1(r_1) \dots R_m(r_m)$

Here, H is the head-daughter of the phrase, which inherits the head-word h from its parent P. $L_1 \dots L_n$ and $R_1 \dots R_n$ are the left and right sisters of H. Either n or m may be zero, and $n = m$ for unary rules.

The addition of lexical heads leads to an enormous number of potential rules, making direct estimation of $P(RHS|LHS)$ infeasible because of sparse data. Therefore, the generation of the RHS of a rule given the LHS is decomposed into three steps: first the head is generated, then the left and right sisters are generated by independent 0th-order Markov processes. The probability of a rule is thus defined as:

$$(2)\ P(RHS|LHS) = P(L_n(l_n)\dots L_1(l_1)H(h),R_1(r_1)\dots R_m(r_m)|P(h)) = P_h(H|P,h) \times \prod_{i=1}^{m+1} P_r(R_i(r_i)|P,h,H,d(i)) \times \prod_{i=1}^{n+1} P_l(L_i(l_i)|P,h,H,d(i))$$

Here, P_h is the probability of generating the head, P_l and P_r are the probabilities of generating the left and right sister respectively. $L_{m+1}(l_{m+1})$ and $R_{m+1}(r_{m+1})$ are defined as stop categories which indicate when to stop generating sisters. $d(i)$ is a distance measure, a function of the length of the surface string between the head and the previously generated sister.

Collins' Model 2 further refines the initial model by incorporating the complement/adjunct distinction and subcategorization frames. The generative process is enhanced to include a probabilistic choice of left and right subcategorization frames. The probability of a rule is now:

$$(3)\ P_h(H|P,h) \times P_{lc}(LC|P,H,h) \times P_{rc}(RC|P,H,h) \times \prod_{i=1}^{m+1} P_r(R_i(r_i)|P,h,H,d(i),RC) \times \prod_{i=1}^{n+1} P_l(L_i(l_i)|P,h,H,d(i),LC)$$

Here, *LC* and *RC* are left and right subcat frames, multisets specifying the complements that the head requires in its left or right sister. The subcat requirements are added to the conditioning context. As complements are generated, they are removed from the appropriate subcat multiset.

5 Experiment 1: Unlexicalized Model

5.1 Method

This experiment was designed to compare the performance of the unlexicalized baseline model on four different datasets, created by the tree transformations described in Section 3: compounds expanded (Exp), compounds contracted (Cont), compounds expanded with coordination raised (Exp+CR), and compounds contracted with coordination raised (Cont+CR).

We used BitPar (Schmid, 2004) for our unlexicalized experiments. BitPar is a parser based on a bit-vector implementation of the CKY algorithm. A grammar and lexicon were read off our training set, along with rule frequencies and frequencies for lexical items, based on which BitPar computes the rule probabilities using maximum likelihood estimation. A frequency distribution for POS tags was also read off the training set; this distribution is used by BitPar to tag unknown words in the test data.

Model	LR	LP	CBs	0CB	≤2CB	Tag	Cov
Exp	59.97	58.64	1.74	39.05	73.23	91.00	99.20
Exp+CR	60.75	60.57	1.57	40.77	75.03	91.08	99.09
Cont	64.19	64.61	1.50	46.74	76.80	93.30	98.48
Cont+CR	**66.11**	**65.55**	1.39	46.99	78.95	93.22	97.94

Table 1: Results for unlexicalized models (sentences ≤40 words); each model performed its own POS tagging.

All models were evaluated using standard Parseval measures of labeled recall (LR), labeled precision (LP), average crossing brackets (CBs), zero crossling brackets (0CB), and two or less crossing brackets (≤2CB). We also report tagging accuracy (Tag), and coverage (Cov).

5.2 Results

The results for the unlexicalized model are shown in Table 1 for sentences of length ≤40 words. We find that contracting compounds increases parsing performance substantially compared to expanding compounds, raising labeled recall from around 60% to around 64% and labeled precision from around 59% to around 65%. The results show that raising coordination is also beneficial; it increases precision and recall by 1–2%, both for expanded and for non-expanded compounds.

Note that these results were obtained by uniformly applying coordination raising during evaluation, so as to make all models comparable. For the Exp and Cont models, the parsed output and the gold standard files were first converted by raising coordination and then the evaluation was performed.

5.3 Discussion

The disappointing performance obtained for the expanded compound models can be partly attributed to the increase in the number of grammar rules (11,704 expanded vs. 10,299 contracted) and POS tags (24 expanded vs. 11 contracted) associated with that transformation.

However, a more important point observation is that the two compound models do not yield comparable results, since an expanded compound has more brackets than a contracted one. We attempted to address this problem by collapsing the compounds for evaluation purposes (as described in Section 3). For example, (5) would be contracted to (4). However, this approach only works if we are certain that the model is tagging the right words as compounds. Unfortunately, this is rarely the case. For example, the model outputs:

(6) (NCmp (N jours) (N commerçants))

But in the gold standard file, *jours* and *commerçants* are two distinct NPs. Collapsing the compounds therefore leads to length mismatches in the test data. This problem occurs frequently in the test set, so that such an evaluation becomes pointless.

6 Experiment 2: Lexicalized Models

6.1 Method

Parsing We now compare a series of lexicalized parsing models against the unlexicalized baseline established in the previous experiment. Our is was to test if French behaves like English in that lexicalization improves parsing performance, or like German, in that lexicalization has only a small effect on parsing performance.

The lexicalized parsing experiments were run using Dan Bikel's probabilistic parsing engine (Bikel, 2002) which in addition to replicating the models described by Collins (1997) also provides a convenient interface to develop corresponding parsing models for other languages.

Lexicalization requires that each rule in a grammar has one of the categories on its right hand side annotated as the head. These head rules were constructed based on the FTB annotation guidelines (provided along with the dataset), as well as by using heuristics, and were optimized on the development set. Collins' Model 2 incorporates a complement/adjunct distinction and probabilities over subcategorization frames. Complements were marked in the training phase based on argument identification rules, tuned on the development set.

Part of speech tags are generated along with the words in the models; parsing and tagging are fully integrated. To achieve this, Bikel's parser requires a mapping of lexical items to orthographic/morphological word feature vectors. The features implemented (capitalization, hyphenation, inflection, derivation, and compound) were again optimized on the development set.

Like BitPar, Bikel's parser implements a probabilistic version of the CKY algorithm. As with normal CKY, even though the model is defined in a top-down, generative manner, decoding proceeds bottom-up. To speed up decoding, the algorithm implements beam search. Collins uses a beam width of 10^4, while we found that a width of 10^5 gave us the best coverage vs. parsing speed trade-off.

Label	FTB	PTB	Negra	Label	FTB	PTB	Negra
SENT	5.84	2.22	4.55	VPpart	2.51	–	–
Ssub	4.41	–	–	VN	1.76	–	–
Sint	3.44	–	–	PP	2.10	2.03	3.08
Srel	3.92	–	–	NP	2.45	2.20	3.08
VP	–	2.32	2.59	AdvP	2.24	–	2.08
VPinf	3.07	–	–	AP	1.34	–	2.22

Table 2: Average number of daughter nodes per constituents in three treebanks

Flatness As already pointed out in Section 2.1, the FTB uses a flat annotation scheme. This can be quantified by computing the average number of daughters for each syntactic category in the FTB, and comparing them with the figures available for PTB and Negra (Dubey and Keller, 2003). This is done in Table 2. The absence of sentence-internal VPs explains the very high level of flatness for the sentential category SENT (5.84 daughters), compared to the PTB (2.44), and even to Negra, which is also very flat (4.55 daughters). The other sentential categories Ssub (subordinate clauses), Srel (relative clause), and Sint (interrogative clause) are also very flat. Note that the FTB uses VP nodes only for non-finite subordinate clauses: VPinf (infinitival clause) and VPpart (participle clause); these categories are roughly comparable in flatness to the VP category in the PTB and Negra. For NP, PPs, APs, and AdvPs the FTB is roughly as flat as the PTB, and somewhat less flat than Negra.

Sister-Head Model To cope with the flatness of the FTB, we implemented three additional parsing models. First, we implemented Dubey and Keller's (2003) sister-head model, which extends Collins' base NP model to all syntactic categories. This means that the probability function P_r in equation (2) is no longer conditioned on the head but instead on its previous sister, yielding the following definition for P_r (and by analogy P_l):

(4) $P_r(R_i(r_i)|P, R_{i-1}(r_{i-1}), d(i))$

Dubey and Keller (2003) argue that this implicitly adds binary branching to the grammar, and therefore provides a way of dealing with flat annotation (in Negra and in the FTB, see Table 2).

Bigram Model This model, inspired by the approach of Collins et al. (1999) for parsing the Prague Dependency Treebank, builds on Collins' Model 2 by implementing a 1st order Markov assumption for the generation of sister non-terminals. The latter are now conditioned, not only on their head, but also on the previous sister. The probability function for P_r (and by analogy P_l) is now:

(5) $P_r(R_i(r_i)|P, h, H, d(i), R_{i-1}, RC)$

Model	LR	LP	CBs	0CB	≤2CB	Tag	Cov
Model 1	80.35	79.99	0.78	65.22	89.46	96.86	99.68
Model 2	80.49	79.98	0.77	64.85	90.10	96.83	99.68
SisterHead	80.47	80.56	0.78	64.96	89.34	96.85	99.57
Bigram	**81.15**	**80.84**	0.74	65.21	90.51	96.82	99.46
BigramFlat	80.30	80.05	0.77	64.78	89.13	96.71	99.57

Table 3: Results for lexicalized models (sentences ≤40 words); each model performed its own POS tagging; all lexicalized models used the Cont+CR data set

The intuition behind this approach is that the model will learn that the stop symbol is more likely to follow phrases with many sisters. Finally, we also experimented with a third model (BigramFlat) that applies the bigram model only for categories with high degrees of flatness (SENT, Srel, Ssub, Sint, VPinf, and VPpart).

6.2 Results

Constituency Evaluation The lexicalized models were tested on the Cont+CR data set, i.e., compounds were contracted and coordination was raised (this is the configuration that gave the best performance in Experiment 1).

Table 3 shows that all lexicalized models achieve a performance of around 80% recall and precision, i.e., they outperform the best unlexicalized model by at least 14% (see Table 1). This is consistent with what has been reported for English on the PTB.

Collins' Model 2, which adds the complement/adjunct distinction and subcategorization frames achieved only a very small improvement over Collins' Model 1, which was not statistically significant using a χ^2 test. It might well be that the annotation scheme of the FTB does not lend itself particularly well to the demands of Model 2. Moreover, as Collins (1997) mentions, some of the benefits of Model 2 are already captured by inclusion of the distance measure.

A further small improvement was achieved using Dubey and Keller's (2003) sister-head model; however, again the difference did not reach statistical significance. The bigram model, however, yielded a statistically significant improvement over Collins' Model 1 (recall $\chi^2 = 3.91$, $df = 1$, $p \leq .048$; precision $\chi^2 = 3.97$, $df = 1$, $p \leq .046$). This is consistent with the findings of Collins et al. (1999) for Czech, where the bigram model upped dependency accuracy by about 0.9%, as well as for English where Charniak (2000) reports an increase in F-score of approximately 0.3%. The BigramFlat model, which applies the bigram model to only those labels which have a high degree of flatness, performs at roughly the same level as Model 1.

Model	LR	LP	CBs	0CB	≤2CB	Tag	Cov
Exp+CR	65.50	64.76	1.49	42.36	77.48	100.0	97.83
Cont+CR	69.35	67.93	1.34	47.43	80.25	100.0	96.97
Model1	81.51	81.43	0.78	64.60	89.25	98.54	99.78
Model2	81.69	81.59	0.78	63.84	89.69	98.55	99.78
SisterHead	81.08	81.56	0.79	64.35	89.57	98.51	99.57
Bigram	**81.78**	**81.91**	0.78	64.96	89.12	98.81	99.67
BigramFlat	81.14	81.19	0.81	63.37	88.80	98.80	99.67

Table 4: Results for lexicalized and unlexicalized models (sentences ≤40 words) with correct POS tags supplied; all lexicalized models used the Cont+CR data set

The models in Tables 1 and 3 implemented their own POS tagging. Tagging accuracy was 91–93% for BitPar (unlexicalized models) and around 96% for the word-feature enhanced tagging model of the Bikel parser (lexicalized models). POS tags are an important cue for parsing. To gain an upper bound on the performance of the parsing models, we reran the experiments by providing the correct POS tag for the words in the test set. While BitPar always uses the tags provided, the Bikel parser only uses them for words whose frequency is less than the unknown word threshold. As Table 4 shows, perfect tagging increased parsing performance in the lexicalized models by around 3%. This shows that the poor POS tagging performed by BitPar is one of the reasons of the poor performance of the lexicalized models. The impact of perfect tagging is less drastic on the lexicalized models (around 1% increase). However, our main finding, viz., that lexicalized models outperform unlexicalized models considerable on the FTB, remains valid, even with perfect tagging.[3]

Dependency Evaluation We also evaluated our models using dependency measures, which have been argued to be more annotation-neutral than Parseval. Lin (1995) notes that labeled bracketing scores are more susceptible to cascading errors, where one incorrect attachment decision causes the scoring algorithm to count more than one error.

The gold standard and parsed trees were converted into dependency trees using the algorithm described by Lin (1995). Dependency accuracy is defined as the ratio of correct dependencies over the total number of dependencies in a sentence. (Note that this is an unlabeled dependency measure.) Dependency accuracy and constituency F-score are shown

[3]It is important to note that the Collins model has a range of other features that set it apart from a standard unlexicalized PCFG (notably Markovization), as discussed in Section 4.2. It is therefore likely that the gain in performance is not attributable to lexicalization alone.

Model	Dependency	F-score
Cont+CR	73.09	65.83
Model 2	83.96	80.23
SisterHead	84.00	80.51
Bigram	**84.20**	**80.99**

Table 5: Dependency vs. constituency scores for lexicalized and unlexicalized models

in Table 5 for the most relevant FTB models. (F-score is computed as the geometric mean of labeled recall and precision.)

Numerically, dependency accuracies are higher than constituency F-scores across the board. However, the effect of lexicalization is the same on both measures: for the FTB, a gain of 11% in dependency accuracy is observed for the lexicalized model.

7 Experiment 3: Crosslinguistic Comparison

The results reported in Experiments 1 and 2 shed some light on the role of lexicalization for parsing French, but they are not strictly comparable to the results that have been reported for other languages. This is because the treebanks available for different languages typically vary considerably in size: our FTB training set was about 8,500 sentences large, while the standard training set for the PTB is about 40,000 sentences in size, and the Negra training set used by Dubey and Keller (2003) comprises about 18,600 sentences. This means that the differences in the effect of lexicalization that we observe could be simply due to the size of the training set: lexicalized models are more susceptible to data sparseness than unlexicalized ones.

We therefore conducted another experiment in which we applied Collins' Model 2 to subsets of the PTB that were comparable in size to our FTB data sets. We combined sections 02–05 and 08 of the PTB (8,345 sentences in total) to form the training set, and the first 1,000 sentences of section 23 to form our test set. As a baseline model, we also run an unlexicalized PCFG on the same data sets. For comparison with Negra, we also include the results of Dubey and Keller (2003): they report the performance of Collins' Model 1 on a data set of 9,301 sentences and a test set of 1,000 sentences, which are comparable in size to our FTB data sets.[4]

The results of the crosslinguistic comparison are shown in Table 6.[5] We conclude that the effect of

Corpus	Model	LR	LP	CBs	0CB	$\leq$2CB
FTB	Cont+CR	66.11	65.55	1.39	46.99	78.95
	Model 2	79.20	78.58	0.83	63.33	89.23
PTB	Unlex	72.79	75.23	2.54	31.56	58.98
	Model 2	86.43	86.79	1.17	57.80	82.44
Negra	Unlex	69.64	67.27	1.12	54.21	82.84
	Model 1	68.33	67.32	0.83	60.43	88.78

Table 6: The effect of lexicalization on different corpora for training sets of comparable size (sentences $\leq$40 words)

lexicalization is stable even if the size of the training set is held constant across languages: For the FTB we find that lexicalization increases F-score by around 13%. Also for the PTB, we find an effect of lexicalization of about 14%. For the German Negra treebank, however, the performance of the lexicalized and the unlexicalized model are almost indistinguishable. (This is true for Collins' Model 1; note that Dubey and Keller (2003) do report a small improvement for the lexicalized sister-head model.)

8 Related Work

We are not aware of any previous attempts to build a probabilistic, treebank-trained parser for French. However, there is work on chunking for French. The group who built the French Treebank (Abeillé et al., 2000) used a rule-based chunker to automatically annotate the corpus with syntactic structures, which were then manually corrected. They report an unlabeled recall/precision of 94.3/94.2% for opening brackets and 92.2/91.4% for closing brackets, and a label accuracy of 95.6%. This result is not comparable to our results for full parsing.

Giguet and Vergne (1997) present use a memory-based learner to predict chunks and dependencies between chunks. The system is evaluated on texts from *Le Monde* (different from the FTB texts). Results are only reported for verb-object dependencies, for which recall/precision is 94.04/96.39%. Again, these results are not comparable to ours, which were obtained using a different corpus, a different dependency scheme, and for a full set of dependencies.

9 Conclusions

In this paper, we provided the first probabilistic, treebank-trained parser for French. In Experiment 1, we established an unlexicalized baseline model, which yielded a labeled precision and recall of about 66%. We experimented with a number of tree transformation that take account of the peculiarities of the annotation of the French Tree-

[4]Dubey and Keller (2003) report only F-scores for the reduced data set (see their Figure 1); the other scores were provided by Amit Dubey. No results for Model 2 are available.

[5]For this experiments, the same POS tagging model was applied to the PTB and the FTB data, which is why the FTB figures are slightly lower than in Table 3.

bank; the best performance was obtained by raising coordination and contracting compounds (which have internal structure in the FTB). In Experiment 2, we explored a range of lexicalized parsing models, and found that lexicalization improved parsing performance by up to 15%: Collins' Models 1 and 2 performed at around 80% LR and LP. No significant improvement could be achieved by switching to Dubey and Keller's (2003) sister-head model, which has been claimed to be particularly suitable for treebanks with flat annotation, such as the FTB. A small but significant improvement (to 81% LR and LP) was obtained by a bigram model that combines features of the sister-head model and Collins' model.

These results have important implications for crosslinguistic parsing research, as they allow us to tease apart language-specific and annotation-specific effects. Previous work for English (e.g., Magerman, 1995; Collins, 1997) has shown that lexicalization leads to a sizable improvement in parsing performance. English is a language with non-flexible word order and with a treebank with a non-flat annotation scheme (see Table 2). Research on German (Dubey and Keller, 2003) showed that lexicalization leads to no sizable improvement in parsing performance for this language. German has a flexible word order and a flat treebank annotation, both of which could be responsible for this counter-intuitive effect. The results for French presented in this paper provide the missing piece of evidence: they show that French behaves like English in that it shows a large effect of lexicalization. Like English, French is a language with non-flexible word order, but like the German Treebank, the French Treebank has a flat annotation. We conclude that Dubey and Keller's (2003) results for German can be attributed to a language-specific factor (viz., flexible word order) rather than to an annotation-specific factor (viz., flat annotation). We confirmed this claim in Experiment 3 by showing that the effects of lexicalization observed for English, French, and German are preserved if the size of the training set is kept constant across languages.

An interesting prediction follows from the claim that word order flexibility, rather than flatness of annotation, is crucial for lexicalization. A language which has a flexible word order (like German), but a non-flat treebank (like English) should show no effect of lexicalization, i.e., lexicalized models are predicted not to outperform unlexicalized ones. In future work, we plan to test this prediction for Korean, a flexible word order language whose treebank (Penn Korean Treebank) has a non-flat annotation.

References

Abeillé, Anne, Lionel Clement, and Alexandra Kinyon. 2000. Building a treebank for French. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*. Athens.

Bikel, Daniel M. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the 2nd International Conference on Human Language Technology Research*. Morgan Kaufmann, San Francisco.

Bikel, Daniel M. 2004. A distributional analysis of a lexicalized statistical parsing model. In Dekang Lin and Dekai Wu, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Barcelona, pages 182–189.

Bikel, Daniel M. and David Chiang. 2000. Two statistical parsing models applied to the Chinese treebank. In *Proceedings of the 2nd ACL Workshop on Chinese Language Processing*. Hong Kong.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*. Seattle, WA, pages 132–139.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics*. Madrid, pages 16–23.

Collins, Michael, Jan Hajič, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. University of Maryland, College Park.

Dubey, Amit and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo, pages 96–103.

Giguet, Emmanuel and Jacques Vergne. 1997. From part-of-speech tagging to memory-based deep syntactic analysis. In *Proceedings of the International Workshop on Parsing Technologies*. Boston, pages 77–88.

Klein, Dan and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo.

Levy, Roger and Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo.

Lin, Dekang. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Montreal, pages 1420–1425.

Magerman, David. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, MA, pages 276–283.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Schiehlen, Michael. 2004. Annotation strategies for probabilistic parsing in German. In *Proceedings of the 20th International Conference on Computational Linguistics*. Geneva.

Schmid, Helmut. 2004. Effi cient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of the 20th International Conference on Computational Linguistics*. Geneva.

Skut, Wojciech, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the 5th Conference on Applied Natural Language Processing*. Washington, DC.

What to do when lexicalization fails: parsing German with suffix analysis and smoothing

Amit Dubey
University of Edinburgh
Amit.Dubey@ed.ac.uk

Abstract

In this paper, we present an unlexicalized parser for German which employs smoothing and suffix analysis to achieve a labelled bracket F-score of 76.2, higher than previously reported results on the NEGRA corpus. In addition to the high accuracy of the model, the use of smoothing in an unlexicalized parser allows us to better examine the interplay between smoothing and parsing results.

1 Introduction

Recent research on German statistical parsing has shown that lexicalization adds little to parsing performance in German (Dubey and Keller, 2003; Beil et al., 1999). A likely cause is the relative productivity of German morphology compared to that of English: German has a higher type/token ratio for words, making sparse data problems more severe. There are at least two solutions to this problem: first, to use better models of morphology or, second, to make unlexicalized parsing more accurate.

We investigate both approaches in this paper. In particular, we develop a parser for German which attains the highest performance known to us by making use of smoothing and a highly-tuned suffix analyzer for guessing part-of-speech (POS) tags from the input text. Rather than relying on smoothing and suffix analysis alone, we also utilize treebank transformations (Johnson, 1998; Klein and Manning, 2003) instead of a grammar induced directly from a treebank.

The organization of the paper is as follows: Section 2 summarizes some important aspects of our treebank corpus. In Section 3 we outline several techniques for improving the performance of unlexicalized parsing without using smoothing, including treebank transformations, and the use of suffix analysis. We show that suffix analysis is not helpful on the treebank grammar, but it does increase performance if used in combination with the treebank transformations we present. Section 4 describes how smoothing can be incorporated into an unlexicalized grammar to achieve state-of-the-art results in German. Rather using one smoothing algorithm, we use three different approaches, allowing us to compare the relative performance of each. An error analysis is presented in Section 5, which points to several possible areas of future research. We follow the error analysis with a comparison with related work in Section 6. Finally we offer concluding remarks in Section 7.

2 Data

The parsing models we present are trained and tested on the NEGRA corpus (Skut et al., 1997), a hand-parsed corpus of German newspaper text containing approximately 20,000 sentences. It is available in several formats, and in this paper, we use the Penn Treebank (Marcus et al., 1993) format of NEGRA.

The annotation used in NEGRA is similar to that used in the English Penn Treebank, with some differences which make it easier to annotate German syntax. German's *flexible word order* would have required an explosion in long-distance dependencies (LDDs) had annotation of NEGRA more closely resembled that of the Penn Treebank. The NEGRA designers therefore chose to use relatively flat trees, encoding elements of flexible word order using grammatical functions (GFs) rather than LDDs wherever possible.

To illustrate flexible word order, consider the sentences *Der Mann sieht den Jungen* ('The man sees the boy') and *Den Jungen sieht der Mann.* Despite the fact the subject and object are swapped in the second sentence, the meaning of both are essentially the same.[1] The two possible word orders are disambiguated by the use of the nominative case for the subject (marked by the article *der*) and the accusative case for the object (marked by *den)* rather than their position in the sentence.

Whenever the subject appears after the verb, the non-standard position may be annotated using a long-distance dependency (LDD). However, as mentioned above, this information can also be retrieved from the grammatical function of the respective noun phrases: the GFs of the two NPs above would be 'subject' and 'accusative object' regardless of their position in the sentence. These labels may therefore be used to recover the underlying dependencies without having to resort to LDDs. This is the approach used in NEGRA. It does have limitations: it is only possible to use GF labels instead of LDDs when all the nodes of interest are dominated by the same parent. To maximize cases where all necessary nodes are dominated by the same parent, NEGRA uses flat 'dependency-style' rules. For example, there is no VP node when there is no overt auxiliary verb. category. Under the NEGRA annotation scheme, the first sentence above would have a rule S$\rightarrow$NP-SB VVFIN NP-OA and the second, S$\rightarrow$NP-OA VVFIN NP-SB, where SB denotes subject and OA denotes accusative object.

3 Parsing with Grammatical Functions

3.1 Model

As explained above, this paper focuses on unlexicalized grammars. In particular, we make use of probabilistic context-free grammars (PCFGs; Booth (1969)) for our experiments. A PCFG assigns each context-free rule LHS $\rightarrow$ RHS a conditional probability $P_r(\text{RHS}|\text{LHS})$. If a parser were to be given POS tags as input, this would be the only distribution required. However, in this paper we are concerned with the more realistic problem of accepting text as input. Therefore, the parser also needs a probability distribution $P_w(w|\text{LHS})$ to generate words. The probability of a tree is calculated by multiplying the probabilities all the rules and words generated in the derivation of the tree.

The rules are simply read out from the treebank, and the probabilities are estimated from the frequency of rules in the treebank. More formally:

$$P_r(\text{RHS}|\text{LHS}) = \frac{c(\text{LHS} \rightarrow \text{RHS})}{c(\text{LHS})} \qquad (1)$$

The probabilities of words given tags are similarly estimated from the frequency of word-tag co-occurrences:

$$P_w(w|\text{LHS}) = \frac{c(\text{LHS}, w)}{c(\text{LHS})} \qquad (2)$$

To handle unseen or infrequent words, all words whose frequency falls below a threshold Ω are grouped together in an 'unknown word' token, which is then treated like an additional word. For our experiments, we use $\Omega = 10$.

We consider several variations of this simple model by changing both P_r and P_w. In addition to the standard formulation in Equation (1), we consider two alternative variants of P_r. The first is a *Markov* context-free rule (Magerman, 1995; Charniak, 2000). A rule may be turned into a Markov rule by first binarizing it, then making independence assumptions on the new binarized rules. Binarizing the rule $A \rightarrow B_1 \ldots B_n$ results in a number of smaller rules $A \rightarrow B_1 A_{B_1}$, $A_{B_1} \rightarrow B_2 A_{B_1 B_2}$, ..., $A_{B_1 \ldots B_{n-1}} \rightarrow B_n$. Binarization does not change the probability of the rule:

$$\begin{aligned} & P(B_1 \ldots B_n|A) \\ = & \prod_{n}^{i=1} P(\rightarrow B_i|A, B_1, \ldots, B_{i-1}) \end{aligned}$$

Making the 2nd order Markov assumption 'forgets' everything earlier then 2 previous sisters. A rule would now be in the form $A_{B_{i-2}B_{i-1}} \rightarrow B_i A_{B_{i-1}B_i}$, and the probability would be:

$$\begin{aligned} & P(B_1 \ldots B_n|A) \\ \approx & \prod_{n}^{i=1} P(B_i|A, B_{i-2}, B_{i-1}) \end{aligned}$$

[1]Pragmatically speaking, the second sentence has a slightly different meaning. A better translation might be: 'It is the boy the man sees.'

The other rule type we consider are linear precedence/immediate dominance (LP/ID) rules (Gazdar et al., 1985). If a context-free rule can be thought of as a LHS token with an ordered list of tokens on the RHS, then an LP/ID rule can be thought of as a LHS token with a *multiset* of tokens on the RHS together with some constraints on the possible orders of tokens on the RHS. Uszkoreit (1987) argues that LP/ID rules with violatable 'soft' constraints are suitable for modelling some aspects of German word order. This makes a probabilistic formulation of LP/ID rules ideal: probabilities act as soft constraints.

Our treatment of probabilistic LP/ID rules generate children one constituent at a time, conditioning upon the parent and a multiset of previously generated children. Formally, the the probability of the rule is approximated as:

$$\begin{aligned} & P(B_1 \dots B_n|A) \\ \approx \; & \prod_{n}^{i=1} P(B_i|A, \{B_j | j < i\}) \end{aligned}$$

In addition to the two additional formulations of the P_r distribution, we also consider one variant of the P_w distribution, which includes the suffix analysis. It is important to clarify that we only change the handling of uncommon and unknown words; those which occur often are handled as normal. suggested different choices for P_w in the face of unknown words: Schiehlen (2004) suggests using a different unknown word token for capitalized versus uncapitalized unknown words (German orthography dictates that all common nouns are capitalized) and Levy and Manning (2004) consider inspecting the last letter the unknown word to guess the part-of-speech (POS) tags. Both of these models are relatively impoverished when compared to the approaches of handling unknown words which have been proposed in the POS tagging literature. Brants (2000) describes a POS tagger with a highly tuned suffix analyzer which considers both capitalization and suffixes as long as 10 letters long. This tagger was developed with German in mind, but neither it nor any other advanced POS tagger morphology analyzer has ever been tested with a full parser. Therefore, we take the novel step of integrating this suffix analyzer into the parser for the second P_w distribution.

3.2 Treebank Re-annotation

Automatic treebank transformations are an important step in developing an accurate unlexicalized parser (Johnson, 1998; Klein and Manning, 2003). Most of our transformations focus upon one part of the NEGRA treebank in particular: the GF labels. Below is a list of GF re-annotations we utilise:

Coord GF In NEGRA, a co-ordinated accusative NP rule might look like NP-OA →NP-CJ KON NP-CJ. KON is the POS tag for a conjunct, and CJ denotes the function of the NP is a coordinate sister. Such a rule hides an important fact: the two co-ordinate sisters are also accusative objects. The Coord GF re-annotation would therefore replace the above rule with NP-OA → NP-OA KON NP-OA.

NP case German articles and pronouns are strongly marked for case. However, the grammatical function of all articles is usually NK, meaning noun kernel. To allow case markings in articles and pronouns to 'communicate' with the case labels on the GFs of NPs, we copy these GFs down into the POS tags of articles and pronouns. For example, a rule like NP-OA → ART-NK NN-NK would be replaced by NP-OA →ART-OA NN-NK. A similar improvement has been independently noted by Schiehlen (2004).

PP case Prepositions determine the case of the NP they govern. While the case is often unambiguous (i.e. *für* 'for' always takes an accusative NP), at times the case may be ambiguous. For instance, *in* 'in' may take either an accusative or dative NP. We use the labels -OA, -OD, etc. for unambiguous prepositions, and introduce new categories AD (accusative/dative ambiguous) and DG (dative/genitive ambiguous) for the ambiguous categories. For example, a rule such as PP → P ART-NK NN-NK is replaced with PP →P-AD ART-AD NN-NK if it is headed by the preposition *in.*

SBAR marking German subordinate clauses have a different word order than main clauses. While subordinate clauses can usually be distinguished from main clauses by their GF, there are some GFs which are used in both cases. This transformation adds an SBAR category to explicitly disambiguate these

	No suffix F-score	With suffix F-score
Normal rules	66.3	66.2
LP/ID rules	66.5	66.6
Markov rules	**69.4**	69.1

Table 1: Effect of rule type and suffix analysis.

cases. The transformation does not add any extra nonterminals, rather it replaces rules such as S $\rightarrow$ KOUS NP V NP (where KOUS is a complementizer POS tag) with SBAR $\rightarrow$ KOUS NP V NP.

S GF One may argue that, as far as syntactic disambiguation is concerned, GFs on S categories primarily serve to distinguish main clauses from subordinate clauses. As we have explicitly done this in the previous transformation, it stands to reason that the GF tags on S nodes may therefore be removed without penalty. If the tags are necessary for semantic interpretation, presumably they could be re-inserted using a strategy such as that of Blaheta and Charniak (2000) The last transformation therefore removes the GF of S nodes.

3.3 Method

To allow comparisons with earlier work on NEGRA parsing, we use the same split of training, development and testing data as used in Dubey and Keller (2003). The first 18,602 sentences are used as training data, the following 1,000 form the development set, and the last 1,000 are used as the test set. We remove long-distance dependencies from all sets, and only consider sentences of length 40 or less for efficiency and memory concerns. The parser is given untagged words as input to simulate a realistic parsing task. A probabilistic CYK parsing algorithm is used to compute the Viterbi parse.

We perform two sets of experiments. In the first set, we vary the rule type, and in the second, we report the additive results of the treebank re-annotations described in Section 3.2. The three rule types used in the first set of experiments are standard CFG rules, our version of LP/ID rules, and 2nd order Markov CFG rules. The second battery of experiments was performed on the model with Markov rules.

In both cases, we report PARSEVAL labeled bracket scores (Magerman, 1995), with the brackets labeled by syntactic categories but not grammatical functions. Rather than reporting precision and recall of labelled brackets, we report only the F-score, i.e. the harmonic mean of precision and recall.

	No suffix F-score	With suffix F-score
GF Baseline	69.4	69.1
+Coord GF	70.2	71.5
+NP case	71.1	72.4
+PP case	71.0	72.7
+SBAR	70.9	72.6
+S GF	71.3	**73.1**

Table 2: Effect of re-annotation and suffix analysis with Markov rules.

3.4 Results

Table 1 shows the effect of rule type choice, and Table 2 lists the effect of the GF re-annotations. From Table 1, we see that Markov rules achieve the best performance, ahead of both standard rules as well as our formulation of probabilistic LP/ID rules.

In the first group of experiments, suffix analysis marginally lowers performance. However, a different pattern emerges in the second set of experiments. Suffix analysis consistently does better than the simpler word generation probability model.

Looking at the treebank transformations with suffix analysis enabled, we find the coordination re-annotation provides the greatest benefit, boosting performance by 2.4 to 71.5. The NP and PP case re-annotations together raise performance by 1.2 to 72.7. While the SBAR annotation slightly lowers performance, removing the GF labels from S nodes increased performance to 73.1.

3.5 Discussion

There are two primary results: first, although LP/ID rules have been suggested as suitable for German's flexible word order, it appears that Markov rules actually perform better. Second, adding suffix analysis provides a clear benefit, but only after the inclusion of the Coord GF transformation.

While the SBAR transformation slightly reduces performance, recall that we argued the S GF transformation only made sense if the SBAR transformation is already in place. To test if this was indeed the case, we re-ran the final experiment, but excluded the SBAR transformation. We did indeed find that applying S GF without the SBAR transformation reduced performance.

4 Smoothing & Search

With the exception of DOP models (Bod, 1995), it is uncommon to smooth unlexicalized grammars. This is in part for the sake of simplicity: unlexicalized grammars are interesting because they are simple to estimate and parse, and adding smoothing makes both estimation and parsing nearly as complex as with fully lexicalized models. However, because lexicalization adds little to the performance of German parsing models, it is therefore interesting to investigate the impact of smoothing on unlexicalized parsing models for German.

Parsing an unsmoothed unlexicalized grammar is relatively efficient because the grammar constraints the search space. As a smoothed grammar does not have a constrained search space, it is necessary to find other means to make parsing faster. Although it is possible to efficiently compute the Viterbi parse (Klein and Manning, 2002) using a smoothed grammar, the most common approach to increase parsing speed is to use some form of beam search (cf. Goodman (1998)), a strategy we follow here.

4.1 Models

We experiment with three different smoothing models: the modified Witten-Bell algorithm employed by Collins (1999), the modified Kneser-Ney algorithm of Chen and Goodman (1998) the smoothing algorithm used in the POS tagger of Brants (2000). All are variants of linear interpolation, and are used with 2^{nd} order Markovization. Under this regime, the probability of adding the i^{th} child to $\mathrm{A} \rightarrow \mathrm{B}_1 \ldots \mathrm{B}_n$ is estimated as

$$\begin{aligned} & P(B_i|A,B_{i-1},B_{i-2}) \\ = \; & \lambda_1 P(B_i|A,B_{i-1},B_{i-2}) + \\ & \lambda_2 P(B_i|A,B_{i-1}) + \lambda_3 P(B_i|A) + \lambda_4 P(B_i) \end{aligned}$$

The models differ in how the λ's are estimated. For both the Witten-Bell and Kneser-Ney algorithms, the λ's are a function of the context A, B_{i-2}, B_{i-1}. By contrast, in Brants' algorithm the λ's are constant for all possible contexts. As both the Witten-Bell and Kneser-Ney variants are fairly well known, we do not describe them further. However, as Brants' approach (to our knowledge) has not been used elsewhere, and because it needs to be modified for our purposes, we show the version of the algorithm we use in Figure 1.

```
$\lambda_1, \lambda_2, \lambda_3 \leftarrow 0$
for each trigram $x_1, x_2, x_3$ with $c(x_1, x_2, x_3) > 0$
    $d_3 \leftarrow \begin{cases} \frac{c(x_i, x_{i-1}, x_{i-2}) - 1}{c(x_{i-1}, x_{i-2}) - 1} & \text{if } c(x_{i-1}, x_{i-2}) > 1 \\ 0 & \text{if } c(x_{i-1}, x_{i-2}) = 1 \end{cases}$
    $d_2 \leftarrow \begin{cases} \frac{c(x_i, x_{i-1}) - 1}{c(x_{i-1}) - 1} & \text{if } c(x_{i-1}) > 1 \\ 0 & \text{if } c(x_{i-1}) = 1 \end{cases}$
    $d_1 \leftarrow \frac{c(x_i) - 1}{N - 1}$
    if $d_3 = \max d_1, d_2, d_3$ then
        $\lambda_3 \leftarrow \lambda_3 + c(x_i, x_{i-1}, x_{i-2})$
    elseif $d_2 = \max d_1, d_2, d_3$ then
        $\lambda_2 \leftarrow \lambda_2 + c(x_i, x_{i-1}, x_{i-2})$
    else
        $\lambda_1 \leftarrow \lambda_1 + c(x_i, x_{i-1}, x_{i-2})$
end
$\lambda_1 \leftarrow \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda + 3}$
$\lambda_2 \leftarrow \frac{\lambda_2}{\lambda_1 + \lambda_2 + \lambda + 3}$
$\lambda_3 \leftarrow \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda + 3}$
```

Figure 1: Smoothing estimation based on the Brants (2000) approach for POS tagging.

4.2 Method

The purpose of this is experiment is not only to improve parsing results, but also to investigate the overall effect of smoothing on parse accuracy. Therefore, we do not simply report results with the best model from Section 3. Rather, we re-do each modification in Section 3 with both search strategies (Viterbi and beam) in the unsmoothed case, and with all three smoothing algorithms with beam search. The beam has a variable width, which means an arbitrary number of edges may be considered, as long as their probability is within 4×10^{-3} of the best edge in a given span.

4.3 Results

Table 3 summarizes the results. The best result in each column is italicized, and the overall best result

	No Smoothing Viterbi	No Smoothing Beam	Brants Beam	Kneser-Ney Beam	Witten-Bell Beam
GF Baseline	69.1	70.3	72.3	72.6	72.3
+Coord GF	71.5	72.7	75.2	75.4	74.5
+NP case	72.4	*73.3*	76.0	76.1	75.6
+PP case	72.7	73.2	76.1	*76.2*	*75.7*
+SBAR	72.6	73.1	**76.3**	76.0	75.3
+S GF Removal	*73.1*	72.6	75.7	75.3	75.1

Table 3: Effect of various smoothing algorithms.

in shown in bold. The column titled Viterbi reproduces the second column of Table 2 whereas the column titled Beam shows the result of re-annotation using beam search, but no smoothing. The best result with beam search is 73.3, slightly higher than without beam search.

Among smoothing algorithms, the Brants approach yields the highest results, of 76.3, with the modified Kneser-Ney algorithm close behind, at 76.2. The modified Witten-Bell algorithm achieved an F-score of 75.7.

4.4 Discussion

Overall, the best-performing model, using Brants smoothing, achieves a labelled bracketing F-score of 76.2, higher than earlier results reported by Dubey and Keller (2003) and Schiehlen (2004).

It is surprisingly that the Brants algorithm performs favourably compared to the better-known modified Kneser-Ney algorithm. This might be due to the heritage of the two algorithms. Kneser-Ney smoothing was designed for language modelling, where there are tens of thousands or hundreds of thousands of tokens having a Zipfian distribution. With all transformations included, the nonterminals of our grammar did have a Zipfian marginal distribution, but there were only several hundred tokens. The Brants algorithm was specifically designed for distributions with fewer tokens.

Also surprising is the fact that each smoothing algorithm reacted differently to the various treebank transformations. It is obvious that the choice of search and smoothing algorithm add *bias* to the final result. However, our results indicate that the choice of search and smoothing algorithm also add a degree of *variance* as improvements are added to the parser. This is worrying: at times in the literature, details of search or smoothing are left out (e.g. Charniak (2000)). Given the degree of variance due to search and smoothing, it raises the question if it is in fact possible to reproduce such results without the necessary details.[2]

5 Error Analysis

While it is uncommon to offer an error analysis for probabilistic parsing, Levy and Manning (2003) argue that a careful error classification can reveal possible improvements. Although we leave the implementation of any improvements to future research, we do discuss several common errors. Because the parser with Brants smoothing performed best, we use that as the basis of our error analysis.

First, we found that POS tagging errors had a strong effect on parsing results. This is surprising, given that the parser is able to assign POS tags with a high degree of accuracy. POS tagging results are comparable to the best stand-alone POS taggers, achieving results of 97.1% on the test set, matching the performance of the POS tagger described by Brants (2000) When GF labels are included (e.g. considering ART-SB instead of just ART), tagging accuracy falls to 90.1%. To quantify the effect of POS tagging errors, we re-parsed with correct POS tags (rather than letting the parser guess the tags), and found that labelled bracket F-scores increase from 76.3 to 85.2. A manual inspection of 100 sentences found that GF mislabelling can accounts for at most two-thirds of the mistakes due to POS tags. Over one third was due to genuine POS tagging errors. The most common problem was verb mistagging: they are either confused with adjectives (both

[2]As an anonymous reviewer pointed out, it is not always straightforward to reproduce statistical parsing results even when the implementation details *are* given (Bikel, 2004).

Model	LB F-score
This paper	**76.3**
Dubey and Keller (2003)	74.1
Schiehlen (2004)	71.1

Table 4: Comparison with previous work.

take the common -en suffix), or the tense was incorrect. Mistagged verb are a serious problem: it entails an entire clause is parsed incorrectly. Verb mistagging is also a problem for other languages: Levy and Manning (2003) describe a similar problem in Chinese for noun/verb ambiguity. This problem might be alleviated by using a more detailed model of morphology than our suffix analyzer provides.

To investigate pure parsing errors, we manually examined 100 sentences which were incorrectly parsed, but which nevertheless were assigned the correct POS tags. Incorrect modifier attachment accounted for for 39% of all parsing errors (of which 77% are due to PP attachment alone). Misparsed coordination was the second most common problem, accounting for 15% of all mistakes. Another class of error appears to be due to Markovization. The boundaries of VPs are sometimes incorrect, with the parser attaching dependents directly to the S node rather than the VP. In the most extreme cases, the VP had no verb, with the main verb heading a subordinate clause.

6 Comparison with Previous Work

Table 4 lists the result of the best model presented here against the earlier work on NEGRA parsing described in Dubey and Keller (2003) and Schiehlen (2004). Dubey and Keller use a variant of the lexicalized Collins (1999) model to achieve a labelled bracketing F-score of 74.1%. Schiehlen presents a number of unlexicalized models. The best model on labelled bracketing achieves an F-score of 71.8%.

The work of Schiehlen is particularly interesting as he also considers a number of transformations to improve the performance of an unlexicalized parser. Unlike the work presented here, Schiehlen does not attempt to perform any suffix or morphological analysis of the input text. However, he does suggest a number of treebank transformations. One such transformation is similar to one we prosed here, the NP case transformation. His implementation is different from ours: he annotates the case of pronouns and common nouns, whereas we focus on articles and pronouns (articles are pronouns are more strongly marked for case than common nouns). The remaining transformations we present are different from those Schiehlen describes; it is possible that an even better parser may result if all the transformations were combined.

Schiehlen also makes use of a morphological analyzer tool. While this includes more complete information about German morphology, our suffix analysis model allows us to integrate morphological ambiguities into the parsing system by means of lexical generation probabilities.

Levy and Manning (2004) also present work on the NEGRA treebank, but are primarily interested in long-distance dependencies, and therefore do not report results on local dependencies, as we do here.

7 Conclusions

In this paper, we presented the best-performing parser for German, as measured by labelled bracket scores. The high performance was due to three factors: (i) treebank transformations (ii) an integrated model of morphology in the form of a suffix analyzer and (iii) the use of smoothing in an unlexicalized grammar. Moreover, there are possible paths for improvement: lexicalization could be added to the model, as could some of the treebank transformations suggested by Schiehlen (2004). Indeed, the suffix analyzer could well be of value in a lexicalized model.

While we only presented results on the German NEGRA corpus, there is reason to believe that the techniques we presented here are also important to other languages where lexicalization provides little benefit: smoothing is a broadly-applicable technique, and if difficulties with lexicalization are due to sparse lexical data, then suffix analysis provides a useful way to get more information from lexical elements which were unseen while training.

In addition to our primary results, we also provided a detailed error analysis which shows that PP attachment and co-ordination are problematic for our parser. Furthermore, while POS tagging is highly accurate, the error analysis also shows it does have surprisingly large effect on parsing errors. Because of the strong impact of POS tagging on parsing results, we conjecture that increasing POS tagging accuracy may be another fruitful area for future parsing research.

References

Franz Beil, Glenn Carroll, Detlef Prescher, Stefan Riezler, and Mats Rooth. 1999. Inside-Outside Estimation of a Lexicalized PCFG for German. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, University of Maryland, College Park.

Daniel M. Bikel. 2004. Intricacies of Collins' Parsing Model. *Computational Linguistics*, 30(4).

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st Conference of the North American Chapter of the ACL (NAACL), Seattle, Washington.*, pages 234–240.

Rens Bod. 1995. *Enriching Linguistics with Statistics: Performance Models of Natural Language*. Ph.D. thesis, University of Amsterdam.

Taylor L. Booth. 1969. Probabilistic Representation of Formal Languages. In *Tenth Annual IEEE Symposium on Switching and Automata Theory*, pages 74–81.

Thorsten Brants. 2000. TnT: A statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle.

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of the 1st Conference of North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, WA.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Amit Dubey and Frank Keller. 2003. Parsing German with Sister-head Dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Sapporo, Japan.

Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. 1985. *Generalized Phase Structure Grammar*. Basil Blackwell, Oxford, England.

Joshua Goodman. 1998. *Parsing inside-out*. Ph.D. thesis, Harvard University.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Dan Klein and Christopher D. Manning. 2002. A* Parsing: Fast Exact Viterbi Parse Selection. Technical Report dbpubs/2002-16, Stanford University.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Roger Levy and Christopher D. Manning. 2003. Is it Harder to Parse Chinese, or the Chinese Treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

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*.

David M. Magerman. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, MA.

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.

Micheal Schiehlen. 2004. Annotation Strategies for Probabilistic Parsing in German. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington, DC.

Hans Uszkoreit. 1987. *Word Order and Constituent Structure in German*. CSLI Publications, Stanford, CA.

Detecting Errors in Discontinuous Structural Annotation

Markus Dickinson
Department of Linguistics
The Ohio State University
`dickinso@ling.osu.edu`

W. Detmar Meurers
Department of Linguistics
The Ohio State University
`dm@ling.osu.edu`

Abstract

Consistency of corpus annotation is an essential property for the many uses of annotated corpora in computational and theoretical linguistics. While some research addresses the detection of inconsistencies in positional annotation (e.g., part-of-speech) and continuous structural annotation (e.g., syntactic constituency), no approach has yet been developed for automatically detecting annotation errors in discontinuous structural annotation. This is significant since the annotation of potentially discontinuous stretches of material is increasingly relevant, from treebanks for free-word order languages to semantic and discourse annotation.

In this paper we discuss how the variation n-gram error detection approach (Dickinson and Meurers, 2003a) can be extended to discontinuous structural annotation. We exemplify the approach by showing how it successfully detects errors in the syntactic annotation of the German TIGER corpus (Brants et al., 2002).

1 Introduction

Annotated corpora have at least two kinds of uses: firstly, as training material and as "gold standard" testing material for the development of tools in computational linguistics, and secondly, as a source of data for theoretical linguists searching for analytically relevant language patterns.

Annotation errors and why they are a problem The high quality annotation present in "gold standard" corpora is generally the result of a manual or semi-automatic mark-up process. The annotation thus can contain annotation errors from automatic (pre-)processes, human post-editing, or human annotation. The presence of errors creates problems for both computational and theoretical linguistic uses, from unreliable training and evaluation of natural language processing technology (e.g., van Halteren, 2000; Květŏn and Oliva, 2002, and the work mentioned below) to low precision and recall of queries for already rare linguistic phenomena. Investigating the quality of linguistic annotation and improving it where possible thus is a key issue for the use of annotated corpora in computational and theoretical linguistics.

Illustrating the negative impact of annotation errors on computational uses of annotated corpora, van Halteren et al. (2001) compare taggers trained and tested on the Wall Street Journal (WSJ, Marcus et al., 1993) and the Lancaster-Oslo-Bergen (LOB, Johansson, 1986) corpora and find that the results for the WSJ perform significantly worse. They report that the lower accuracy figures are caused by inconsistencies in the WSJ annotation and that 44% of the errors for their best tagging system were caused by "inconsistently handled cases."

Turning from training to evaluation, Padro and Marquez (1998) highlight the fact that the true accuracy of a classifier could be much better or worse than reported, depending on the error rate of the corpus used for the evaluation. Evaluating two taggers on the WSJ, they find tagging accuracy rates for ambiguous words of 91.35% and 92.82%. Given the estimated 3% error rate of the WSJ tagging (Marcus et al., 1993), they argue that the difference in performance is not sufficient to establish which of the two taggers is actually better.

In sum, corpus annotation errors, especially errors which are inconsistencies, can have a profound impact on the quality of the trained classifiers and the evaluation of their performance. The problem is compounded for syntactic annotation, given the difficulty of evaluating and comparing syntactic structure assignments, as known from the literature on parser evaluation (e.g., Carroll et al., 2002).

The idea that variation in annotation can indicate annotation errors has been explored to detect errors in part-of-speech (POS) annotation (van Halteren, 2000; Eskin, 2000; Dickinson and Meurers, 2003a) and syntactic annotation (Dickinson and Meurers, 2003b). But, as far as we are aware, the research we report on here is the first approach to error detection for the increasing number of annotations which make use of more general graph structures for the syntactic annotation of free word order languages or the annotation of semantic and discourse properties.

Discontinuous annotation and its relevance The simplest kind of annotation is positional in nature, such as the association of a part-of-speech tag with each corpus position. On the other hand, structural annotation such as that used in syntactic treebanks (e.g., Marcus et al., 1993) assigns a syntactic category to a contiguous sequence of corpus positions. For languages with relatively free constituent order, such as German, Dutch, or the Slavic languages, the combinatorial potential of the language encoded in constituency cannot be mapped straightforwardly onto the word order possibilities of those languages. As a consequence, the treebanks that have been created for German (NEGRA, Skut et al., 1997; VERBMOBIL, Hinrichs et al., 2000; TIGER, Brants et al., 2002) have relaxed the requirement that constituents have to be contiguous. This makes it possible to syntactically annotate the language data as such, i.e., without requiring postulation of empty elements as placeholders or other theoretically motivated changes to the data. We note in passing that discontinuous constituents have also received some support in theoretical linguistics (cf., e.g., the articles collected in Huck and Ojeda, 1987; Bunt and van Horck, 1996).

Discontinuous constituents are strings of words which are not necessarily contiguous, yet form a single constituent with a single label, such as the noun phrase *Ein Mann, der lacht* in the German relative clause extraposition example (1) (Brants et al., 2002).[1]

(1)	**Ein**	**Mann**	kommt	,	**der**	**lacht**
	a	man	comes	,	who	laughs

‘A man who laughs comes.’

In addition to their use in syntactic annotation, discontinuous structural annotation is also relevant for semantic and discourse-level annotation—essentially any time that graph structures are needed to encode relations that go beyond ordinary tree structures. Such annotations are currently employed in the mark-up for semantic roles (e.g., Kingsbury et al., 2002) and multi-word expressions (e.g., Rayson et al., 2004), as well as for spoken language corpora or corpora with multiple layers of annotation which cross boundaries (e.g., Blache and Hirst, 2000).

In this paper, we present an approach to the detection of errors in discontinuous structural annotation. We focus on syntactic annotation with potentially discontinuous constituents and show that the approach successfully deals with the discontinuous syntactic annotation found in the TIGER treebank (Brants et al., 2002).

2 The variation n-gram method

Our approach builds on the variation n-gram algorithm introduced in Dickinson and Meurers (2003a,b). The basic idea behind that approach is that a string occurring more than once can occur with different labels in a corpus, which we refer to as *variation*. Variation is caused by one of two reasons: i) *ambiguity*: there is a type of string with multiple possible labels and different corpus occurrences of that string realize the different options, or ii) *error*: the tagging of a string is inconsistent across comparable occurrences.

[1]The ordinary way of marking a constituent with brackets is inadequate for discontinuous constituents, so we instead boldface and underline the words belonging to a discontinuous constituent.

The more similar the context of a variation, the more likely the variation is an error. In Dickinson and Meurers (2003a), contexts are composed of words, and identity of the context is required. The term *variation n-gram* refers to an n-gram (of words) in a corpus that contains a string annotated differently in another occurrence of the same n-gram in the corpus. The string exhibiting the variation is referred to as the *variation nucleus*.

2.1 Detecting variation in POS annotation

In Dickinson and Meurers (2003a), we explore this idea for part-of-speech annotation. For example, in the WSJ corpus the string in (2) is a variation 12-gram since *off* is a variation nucleus that in one corpus occurrence is tagged as a preposition (IN), while in another it is tagged as a particle (RP).[2]

(2) to ward off a hostile takeover attempt by two European shipping concerns

Once the variation n-grams for a corpus have been computed, heuristics are employed to classify the variations into errors and ambiguities. The first heuristic encodes the basic fact that the label assignment for a nucleus is dependent on the context: variation nuclei in long n-grams are likely to be errors. The second takes into account that natural languages favor the use of local dependencies over non-local ones: nuclei found at the fringe of an n-gram are more likely to be genuine ambiguities than those occurring with at least one word of surrounding context. Both of these heuristics are independent of a specific corpus, annotation scheme, or language.

We tested the variation error detection method on the WSJ and found 2495 distinct[3] nuclei for the variation n-grams between the 6-grams and the 224-grams. 2436 of these were actual errors, making for a precision of 97.6%, which demonstrates the value of the long context heuristic. 57 of the 59 genuine ambiguities were fringe elements, confirming that fringe elements are more indicative of a true ambiguity.

[2]To graphically distinguish the variation nucleus within a variation n-gram, the nucleus is shown in grey.

[3]Being distinct means that each corpus position is only taken into account for the longest variation n-gram it occurs in.

2.2 Detecting variation in syntactic annotation

In Dickinson and Meurers (2003b), we decompose the variation n-gram detection for syntactic annotation into a series of runs with different nucleus sizes. This is needed to establish a one-to-one relation between a unit of data and a syntactic category annotation for comparison. Each run detects the variation in the annotation of strings of a specific length. By performing such runs for strings from length 1 to the length of the longest constituent in the corpus, the approach ensures that all strings which are analyzed as a constituent somewhere in the corpus are compared to the annotation of all other occurrences of that string.

For example, the variation 4-gram *from a year earlier* appears 76 times in the WSJ, where the nucleus *a year* is labeled noun phrase (NP) 68 times, and 8 times it is not annotated as a constituent and is given the special label NIL. An example with two syntactic categories involves the nucleus *next Tuesday* as part of the variation 3-gram *maturity next Tuesday*, which appears three times in the WSJ. Twice it is labeled as a noun phrase (NP) and once as a prepositional phrase (PP).

To be able to efficiently calculate all variation nuclei of a treebank, in Dickinson and Meurers (2003b) we make use of the fact that a variation necessarily involves at least one constituent occurrence of a nucleus and calculate the set of nuclei for a window of length i by first finding the constituents of that length. Based on this set, we then find non-constituent occurrences of all strings occurring as constituents. Finally, the variation n-grams for these variation nuclei are obtained in the same way as for POS annotation.

In the WSJ, the method found 34,564 variation nuclei, up to size 46; an estimated 71% of the 6277 non-fringe distinct variation nuclei are errors.

3 Discontinuous constituents

In Dickinson and Meurers (2003b), we argued that null elements need to be ignored as variation nuclei because the variation in the annotation of a null element as the nucleus is largely independent of the local environment. For example, in (3) the null element **EXP** (expletive) can be annotated a. as a sentence (S) or b. as a relative/subordinate clause

(SBAR), depending on the properties of the clause it refers to.

(3) a. For cities losing business to suburban shopping centers , it *EXP* may be a wise business investment [$_S$ * to help * keep those jobs and sales taxes within city limits] .

b. But if the market moves quickly enough , it *EXP* may be impossible [$_{SBAR}$ for the broker to carry out the order] because the investment has passed the specified price .

We found that removing null elements as variation nuclei of size 1 increased the precision of error detection to 78.9%.

Essentially, null elements represent discontinuous constituents in a formalism with a context-free backbone (Bies et al., 1995). Null elements are co-indexed with a non-adjacent constituent; in the predicate argument structure, the constituent should be interpreted where the null element is.

To be able to annotate discontinuous material without making use of inserted null elements, some treebanks have instead relaxed the definition of a linguistic tree and have developed more complex graph annotations. An error detection method for such corpora thus does not have to deal with the problems arising from inserted null elements discussed above, but instead it must function appropriately even if constituents are discontinuously realized.

A technique such as the variation n-gram method is applicable to corpora with a one-to-one mapping between the text and the annotation. For corpora with positional annotation—e.g., part-of-speech annotated corpora—the mapping is trivial given that the annotation consists of one-to-one correspondences between words (i.e., tokens) and labels. For corpora annotated with more complex structural information—e.g., syntactically-annotated corpora—the one-to-one mapping is obtained by considering every interval (continuous string of any length) which is assigned a category label somewhere in the corpus.

While this works for treebanks with continuous constituents, a one-to-one mapping is more complicated to establish for syntactic annotation involving discontinuous constituents (NEGRA, Skut et al., 1997; TIGER, Brants et al., 2002). In order to apply the variation n-gram method to discontinuous constituents, we need to develop a technique which is capable of comparing labels for any set of corpus positions, instead of for any interval.

4 Extending the variation n-gram method

To extend the variation n-gram method to handle discontinuous constituents, we first have to define the characteristics of such a constituent (section 4.1), in other words our units of data for comparison. Then, we can find identical non-constituent (NIL) strings (section 4.2) and expand the context into variation n-grams (section 4.3).

4.1 Variation nuclei: Constituents

For traditional syntactic annotation, a variation nucleus is defined as a contiguous string with a single label; this allows the variation n-gram method to be broken down into separate runs, one for each constituent size in the corpus. For discontinuous syntactic annotation, since we are still interested in comparing cases where the nucleus is the same, we will treat two constituents as having the same size if they consist of the same number of words, regardless of the amount of intervening material, and we can again break the method down into runs of different sizes. The intervening material is accounted for when expanding the context into n-grams.

A question arises concerning the word order of elements in a constituent. Consider the German example (4) (Müller, 2004).

(4)

weil	der Mann	der Frau	das
because	the man$_{nom}$	the woman$_{dat}$	the
Buch	gab.		
book$_{acc}$	gave		

'because the man gave the woman the book.'

The three arguments of the verb *gab* ('give') can be permuted in all six possible ways and still result in a well-formed sentence. It might seem, then, that we would want to allow different permutations of nuclei to be treated as identical. If *das Buch der Frau gab* is a constituent in another sentence, for instance, it should have the same category label as *der Frau das Buch gab*.

Putting all permutations into one equivalence class, however, amounts to stating that all orderings are always the same. But even "free word order" languages are more appropriately called free constituent order; for example, in (4), the argument noun phrases can be freely ordered, but each argument noun phrase is an atomic unit, and in each unit the determiner precedes the noun.

Since we want our method to remain data-driven and order can convey information which might be reflected in an annotation system, we keep strings with different orders of the same words distinct, i.e., ordering of elements is preserved in our method.

4.2 Variation nuclei: Non-constituents

The basic idea is to compare a string annotated as a constituent with the same string found elsewhere—whether annotated as a constituent or not. So we need to develop a method for finding all string occurrences not analyzed as a constituent (and assign them the special category label NIL). Following Dickinson and Meurers (2003b), we only look for non-constituent occurrences of those strings which also occur at least once as a constituent.

But do we need to look for discontinuous NIL strings or is it sufficient to assume only continuous ones? Consider the TIGER treebank examples (5).

(5) a. in diesem Punkt seien **sich** Bonn und
on this point are SELF Bonn and
London nicht **einig** .
London not agreed .

'Bonn and London do not agree on this point.'

b. in diesem Punkt seien **sich** Bonn und
on this point are SELF Bonn and
London offensichtlich **nicht einig** .
London clearly not agreed .

In example (5a), *sich einig* ('SELF agree') forms an adjective phrase (AP) constituent. But in example (5b), that same string is not analyzed as a constituent, despite being in a nearly identical sentence. We would thus like to assign the discontinuous string *sich einig* in (5b) the label NIL, so that the labeling of this string in (5a) can be compared to its occurrence in (5b).

In consequence, our approach should be able to detect NIL strings which are discontinuous—an issue which requires special attention to obtain an algorithm efficient enough to handle large corpora.

Use sentence boundary information The first consideration makes use of the fact that syntactic annotation by its nature respects sentence boundaries. In consequence, we never need to search for NIL strings that span across sentences.[4]

Use tries to store constituent strings The second consideration concerns how we calculate the NIL strings. To find every non-constituent string in the corpus, discontinuous or not, which is identical to some constituent in the corpus, a basic approach would first generate all possible strings within a sentence and then test to see which ones occur as a constituent elsewhere in the corpus. For example, if the sentence is *Nobody died when Clinton lied*, we would see if any of the 31 subsets of strings occur as constituents (e.g., *Nobody*, *Nobody when*, *Clinton lied*, *Nobody when lied*, etc.). But such a generate and test approach clearly is intractable given that it generates generates $2^n - 1$ potential matches for a sentence of n words.

We instead split the task of finding NIL strings into two runs through the corpus. In the first, we store all constituents in the corpus in a trie data structure (Fredkin, 1960), with words as nodes. In the second run through the corpus, we attempt to match the strings in the corpus with a path in the trie, thus identifying all strings occurring as constituents somewhere in the corpus.

Filter out unwanted NIL strings The final consideration removes "noisy" NIL strings from the candidate set. Certain NIL strings are known to be useless for detecting annotation errors, so we should remove them to speed up the variation n-gram calculations. Consider example (6) from the TIGER corpus, where the continuous constituent *die Menschen* is annotated as a noun phrase (NP).

(6) Ohne diese Ausgaben, so die
without these expenses according to the
Weltbank, seien **die Menschen** totes Kapital
world bank are the people dead capital

'According to the world bank, the people are dead capital without these expenses.'

[4]This restriction clearly is syntax specific and other topological domains need to be identified to make searching for NIL strings tractable for other types of discontinuous annotation.

Our basic method of finding NIL strings would detect another occurrence of *die Menschen* in the same sentence since nothing rules out that the other occurrence of *die* in the sentence (preceding *Weltbank*) forms a discontinuous NIL string with *Menschen*. Comparing a constituent with a NIL string that contains one of the words of the constituent clearly goes against the original motivation for wanting to find discontinuous strings, namely that they show variation between different occurrences of a string.

To prevent such unwanted variation, we eliminate occurrences of NIL-labeled strings that overlap with identical constituent strings from consideration.

4.3 Variation n-grams

The more similar the context surrounding a variation nucleus, the more likely it is for a variation in its annotation to be an error. For detecting errors in traditional syntactic annotation (see section 2.2), the context consists of the elements to the left and the right of the nucleus. When nuclei can be discontinuous, however, there can also be *internal context*, i.e., elements which appear between the words forming a discontinuous variation nucleus.

As in our earlier work, an instance of the a priori algorithm is used to expand a nucleus into a longer n-gram by stepwise adding context elements. Where previously it was possible to add an element to the left or the right, we now also have the option of adding it in the middle—as part of the new, internal context. But depending on how we fill in the internal context, we can face a serious tractability problem. Given a nucleus with j gaps within it, we need to potentially expand it in $j + 2$ directions, instead of in just 2 directions (to the right and to the left).

For example, the potential nucleus *was werden* appears as a verb phrase (VP) in the TIGER corpus in the string *was ein Seeufer werden*; elsewhere in the corpus *was* and *werden* appear in the same sentence with 32 words between them. The chances of one of the middle 32 elements matching something in the internal context of the VP is relatively high, and indeed the twenty-sixth word is *ein*. However, if we move stepwise out from the nucleus in order to try to match *was ein Seeufer werden*, the only options are to find *ein* directly to the right of *was* or *Seeufer* directly to the left of *werden*, neither of which occurs, thus stopping the search.

In conclusion, we obtain an efficient application of the a priori algorithm by expanding the context only to elements which are adjacent to an element already in the n-gram. Note that this was already implicitly assumed for the left and the right context.

There are two other efficiency-related issues worth mentioning. Firstly, as with the variation nucleus detection, we limit the n-grams expansion to sentences only. Since the category labels do not represent cross-sentence dependencies, we gain no new information if we find more context outside the sentence, and in terms of efficiency, we cut off what could potentially be a very large search space.[5]

Secondly, the methods for reducing the number of variation nuclei discussed in section 4.2 have the consequence of also reducing the number of possible variation n-grams. For example, in a test run on the NEGRA corpus we allowed identical strings to overlap; this generated a variation nucleus of size 63, with 16 gaps in it, varying between NP and NIL within the same sentence. Fifteen of the gaps can be filled in and still result in variation. The filter for unwanted NIL strings described in the previous section eliminates the NIL value from consideration. Thus, there is no variation and no tractability problem in constructing n-grams.

4.3.1 Generalizing the n-gram context

So far, we assumed that the context added around variation nuclei consists of words. Given that treebanks generally also provide part-of-speech information for every token, we experimented with part-of-speech tags as a less restrictive kind of context. The idea is that it should be possible to find more variation nuclei with comparable contexts if only the part-of-speech tags of the surrounding words have to be identical instead of the words themselves.

As we will see in section 5, generalizing n-gram contexts in this way indeed results in more variation n-grams being found, i.e., increased recall.

4.4 Adapting the heuristics

To determine which nuclei are errors, we can build on the two heuristics from previous research (Dick-

[5]Note that similar sentences which were segmented differently could potentially cause varying n-gram strings not to be found. We propose to treat this as a separate sentence segmentation error detection phase in future work.

inson and Meurers, 2003a,b)—trust long contexts and distrust the fringe—with some modification, given that we have more fringe areas to deal with for discontinuous strings. In addition to the right and the left fringe, we also need to take into account the internal context in a way that maintains the non-fringe heuristic as a good indicator for errors. As a solution that keeps internal context on a par with the way external context is treated in our previous work, we require one word of context around every terminal element that is part of the variation nucleus. As discussed below, this heuristic turns out to be a good predictor of which variations are annotation errors; expanding to the longest possible context, as in Dickinson and Meurers (2003a), is not necessary.

5 Results on the TIGER Corpus

We ran the variation n-grams error detection method for discontinuous syntactic constituents on v. 1 of TIGER (Brants et al., 2002), a corpus of 712,332 tokens in 40,020 sentences. The method detected a total of 10,964 variation nuclei. From these we sampled 100 to get an estimate of the number of errors in the corpus which concern variation. Of these 100, 13 variation nuclei pointed to an error; with this point estimate of .13, we can derive a 95% confidence interval of (0.0641, 0.1959),[6] which means that we are 95% confident that the true number of variation-based errors is between 702 and 2148. The effectiveness of a method which uses context to narrow down the set of variation nuclei can be judged by how many of these variation errors it finds.

Using the non-fringe heuristic discussed in the previous section, we selected the shortest non-fringe variation n-grams to examine. Occurrences of the same strings within larger n-grams were ignored, so as not to artificially increase the resulting set of n-grams.

When the context is defined as identical words, we obtain 500 variation n-grams. Sampling 100 of these and labeling for each position whether it is an error or an ambiguity, we find that 80 out of the 100 samples point to at least one token error. The 95% confidence interval for this point estimate of .80 is (0.7216, 0.8784), so we are 95% confident that the true number of error types is between 361 and 439. Note that this precision is comparable to the estimates for continuous syntactic annotation in Dickinson and Meurers (2003b) of 71% (with null elements) and 78.9% (without null elements).

When the context is defined as identical parts of speech, as described in section 4.3.1, we obtain 1498 variation n-grams. Again sampling 100 of these, we find that 52 out of the 100 point to an error. And the 95% confidence interval for this point estimate of .52 is (0.4221, 0.6179), giving a larger estimated number of errors, between 632 and 926.

Context	Precision	Errors
Word	80%	361–439
POS	52%	632–926

Figure 1: Accuracy rates for the different contexts

Words convey more information than part-of-speech tags, and so we see a drop in precision when using part-of-speech tags for context, but these results highlight a very practical benefit of using a generalized context. By generalizing the context, we maintain a precision rate of approximately 50%, and we substantially increase the recall of the method. There are, in fact, likely twice as many errors when using POS contexts as opposed to word contexts. Corpus annotation projects willing to put in some extra effort thus can use this method of finding variation n-grams with a generalized context to detect and correct more errors.

6 Summary and Outlook

We have described the first method for finding errors in corpora with graph annotations. We showed how the variation n-gram method can be extended to discontinuous structural annotation, and how this can be done efficiently and with as high a precision as reported for continuous syntactic annotation. Our experiments with the TIGER corpus show that generalizing the context to part-of-speech tags increases recall while keeping precision above 50%. The method can thus have a substantial practical benefit when preparing a corpus with discontinuous annotation.

Extending the error detection method to handle

[6]The 95% confidence interval was calculated using the standard formula of $p \pm 1.96\sqrt{\frac{p(1-p)}{n}}$, where p is the point estimate and n the sample size.

discontinuous constituents, as we have done, has significant potential for future work given the increasing number of free word order languages for which corpora and treebanks are being developed.

Acknowledgements We are grateful to George Smith and Robert Langner of the University of Potsdam TIGER team for evaluating the variation we detected in the samples. We would also like to thank the three ACL reviewers for their detailed and helpful comments, and the participants of the OSU CLippers meetings for their encouraging feedback.

References

Ann Bies, Mark Ferguson, Karen Katz and Robert MacIntyre, 1995. *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. University of Pennsylvania.

Philippe Blache and Daniel Hirst, 2000. Multi-level annotation for spoken-language corpora. In *Proceedings of ICSLP-00*. Beijing, China.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius and George Smith, 2002. The TIGER Treebank. In *Proceedings of TLT-02*. Sozopol, Bulgaria.

Harry Bunt and Arthur van Horck (eds.), 1996. *Discontinuous Constituency*. Mouton de Gruyter, Berlin and New York.

John Carroll, Anette Frank, Dekang Lin, Detlef Prescher and Hans Uszkoreit (eds.), 2002. *Proceedings of the LREC Workshop "Beyond PARSEVAL. Towards Improved Evaluation Measures for Parsing Systems"*, Las Palmas, Gran Canaria.

Markus Dickinson and W. Detmar Meurers, 2003a. Detecting Errors in Part-of-Speech Annotation. In *Proceedings of EACL-03*. Budapest, Hungary.

Markus Dickinson and W. Detmar Meurers, 2003b. Detecting Inconsistencies in Treebanks. In *Proceedings of TLT-03*. Växjö, Sweden.

Eleazar Eskin, 2000. Automatic Corpus Correction with Anomaly Detection. In *Proceedings of NAACL-00*. Seattle, Washington.

Edward Fredkin, 1960. Trie Memory. *CACM*, 3(9):490–499.

Erhard Hinrichs, Julia Bartels, Yasuhiro Kawata, Valia Kordoni and Heike Telljohann, 2000. The Tübingen Treebanks for Spoken German, English, and Japanese. In Wolfgang Wahlster (ed.), *Verbmobil: Foundations of Speech-to-Speech Translation*, Springer, Berlin, pp. 552–576.

Geoffrey Huck and Almerindo Ojeda (eds.), 1987. *Discontinuous Constituency*. Academic Press, New York.

Stig Johansson, 1986. *The Tagged LOB Corpus: Users' Manual*. Norwegian Computing Centre for the Humanities, Bergen.

Paul Kingsbury, Martha Palmer and Mitch Marcus, 2002. Adding Semantic Annotation to the Penn TreeBank. In *Proceedings of HLT-02*. San Diego.

Pavel Květŏn and Karel Oliva, 2002. Achieving an Almost Correct PoS-Tagged Corpus. In Petr Sojka, Ivan Kopeček and Karel Pala (eds.), *TSD 2002*. Springer, Heidelberg, pp. 19–26.

M. Marcus, Beatrice Santorini and M. A. Marcinkiewicz, 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Stefan Müller, 2004. Continuous or Discontinuous Constituents? A Comparison between Syntactic Analyses for Constituent Order and Their Processing Systems. *Research on Language and Computation*, 2(2):209–257.

Lluis Padro and Lluis Marquez, 1998. On the Evaluation and Comparison of Taggers: the Effect of Noise in Testing Corpora. In *COLING/ACL-98*.

Paul Rayson, Dawn Archer, Scott Piao and Tony McEnery, 2004. The UCREL Semantic Analysis System. In *Proceedings of the Workshop on Beyond Named Entity Recognition: Semantic labelling for NLP tasks*. Lisbon, Portugal, pp. 7–12.

Wojciech Skut, Brigitte Krenn, Thorsten Brants and Hans Uszkoreit, 1997. An Annotation Scheme for Free Word Order Languages. In *Proceedings of ANLP-97*. Washington, D.C.

Hans van Halteren, 2000. The Detection of Inconsistency in Manually Tagged Text. In Anne Abeillé, Thorsten Brants and Hans Uszkoreit (eds.), *Proceedings of LINC-00*. Luxembourg.

Hans van Halteren, Walter Daelemans and Jakub Zavrel, 2001. Improving Accuracy in Word Class Tagging through the Combination of Machine Learning Systems. *Computational Linguistics*, 27(2):199–229.

High Precision Treebanking
— Blazing Useful Trees Using POS Information —

Takaaki Tanaka,[†] **Francis Bond,**[†] **Stephan Oepen,**[‡] **Sanae Fujita**[†]
[†] {takaaki, bond, fujita}@cslab.kecl.ntt.co.jp
[‡] oe@csli.stanford.edu
[†] NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation
[‡] Universitetet i Oslo and CSLI, Stanford

Abstract

In this paper we present a quantitative and qualitative analysis of annotation in the Hinoki treebank of Japanese, and investigate a method of speeding annotation by using part-of-speech tags. The Hinoki treebank is a Redwoods-style treebank of Japanese dictionary definition sentences. 5,000 sentences are annotated by three different annotators and the agreement evaluated. An average agreement of 65.4% was found using strict agreement, and 83.5% using labeled precision. Exploiting POS tags allowed the annotators to choose the best parse with 19.5% fewer decisions.

1 Introduction

It is important for an annotated corpus that the markup is both correct and, in cases where variant analyses could be considered correct, consistent. Considerable research in the field of word sense disambiguation has concentrated on showing that the annotation of word senses can be done correctly and consistently, with the normal measure being inter-annotator agreement (e.g. Kilgariff and Rosenzweig, 2000). Surprisingly, few such studies have been carried out for syntactic annotation, with the notable exceptions of Brants et al. (2003, p 82) for the German NeGra Corpus and Civit et al. (2003) for the Spanish Cast3LB corpus. Even such valuable and widely used corpora as the Penn TreeBank have not been verified in this way.

We are constructing the Hinoki treebank as part of a larger project in cognitive and computational linguistics ultimately aimed at natural language understanding (Bond et al., 2004). In order to build the initial syntactic and semantic models, we are treebanking the dictionary definition sentences of the most familiar 28,000 words of Japanese and building an ontology from the results.

Arguably the most common method in building a treebank still is manual annotation, annotators (often linguistics students) marking up linguistic properties of words and phrases. In some semi-automated treebank efforts, annotators are aided by POS taggers or phrase-level chunkers, which can propose mark-up for manual confirmation, revision, or extension. As computational grammars and parsers have increased in coverage and accuracy, an alternate approach has become feasible, in which utterances are parsed and the annotator selects the best parse Carter (1997); Oepen et al. (2002) from the full analyses derived by the grammar.

We adopted the latter approach. There were four main reasons. The first was that we wanted to develop a precise broad-coverage grammar in tandem with the treebank, as part of our research into natural language understanding. Treebanking the output of the parser allows us to immediately identify problems in the grammar, and improving the grammar directly improves the quality of the treebank in a mutually beneficial feedback loop (Oepen et al., 2004). The second reason is that we wanted to annotate to a high level of detail, marking not only dependency and constituent structure but also detailed semantic relations. By using a Japanese grammar (JACY: Siegel and Bender, 2002) based on a monostratal theory of grammar (HPSG: Pollard and Sag, 1994) we could simultaneously annotate syntactic and semantic structure without overburdening the annotator. The third reason was that we expected the use of the grammar to aid in enforcing consistency — at the very least all sentences annotated are guaranteed to have well-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 final reason was that the discriminants can be used to update the treebank when the grammar changes, so that the treebank can be improved along with the grammar. This kind of dynamic, discriminant-based treebanking was pioneered in the Redwoods treebank of English (Oepen et al., 2002), so we refer to it as Redwoods-style treebanking.

In the next section, we give some more details about the Hinoki Treebank and the data used to evaluate the parser (§ 2). This is followed by a brief discussion of treebanking using discriminants (§ 3), and an extension to seed the treebanking using existing markup (§ 4). Finally we present the results of our evaluation (§ 5), followed by some discussion and outlines for future research.

2 The Hinoki Treebank

The Hinoki treebank currently consists of around 95,000 annotated dictionary definition and example sentences. The dictionary is the Lexeed Semantic Database of Japanese (Kasahara et al., 2004), which consists of all words with a familiarity greater than or equal to five on a scale of one to seven. This gives 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 81,000 defining sentences in all.

The data used in our evaluation is taken from the first sentence of the definitions of all words with a familiarity greater than six (9,854 sentences). The Japanese grammar JACY was extended until the coverage was over 80% (Bond et al., 2004).

For evaluation of the treebanking we selected 5,000 of the sentences that could be parsed, and divided them into five 1,000 sentence sets (A–E). Definition sentences tend to vary widely in form depending on the part of speech of the word being defined — each set was constructed with roughly the same distribution of defined words, as well as having roughly the same length (the average was 9.9, ranging from 9.5–10.4).

A (simplified) example of an entry (Sense 2 of カーテン*kāten* "curtain: any barrier to communication or vision"), and a syntactic view of its parse are given in Figure 1. There were 6 parses for this definition sentence. The full parse is an HPSG sign, containing both syntactic and semantic information. A view of the semantic information is given in Figure 2[1].

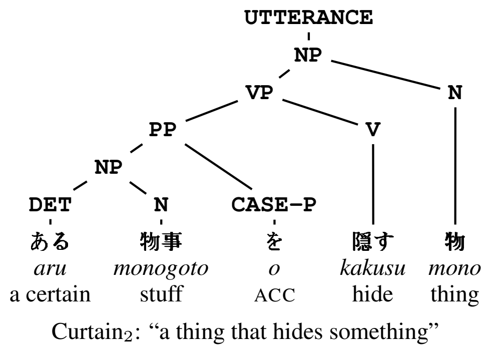


Figure 1: Syntactic View of the Definition of カーテン$_2$ *kāten* "curtain"

$$\begin{array}{lll}
\langle h_0, x_2\ \{ & h_0 : proposition(h_5) & \\
 & h_1 : aru(e_1, x_1, u_0) & \text{"a certain"} \\
 & h_1 : monogoto(x_1) & \text{"stuff"} \\
 & h_2 : u\_def(x_1, h_1, h_6) & \\
 & h_5 : kakusu(e_2, x_2, x_1) & \text{"hide"} \\
 & h_3 : mono(x_2) & \text{"thing"} \\
 & h_4 : u\_def(x_2, h_3, h_7)\}\rangle &
\end{array}$$

Figure 2: Semantic View of the Definition of カーテン$_2$ *kāten* "curtain"

The semantic view shows some ambiguity has been resolved that is not visible in the purely syntactic view. In Japanese, relative clauses can have gapped and non-gapped readings. In the gapped reading (selected here), 物 *mono* "thing" is the subject of 隠す *kakusu* "hide". In the non-gapped reading there is some unspecified 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).

[1]The semantic representation used is Minimal Recursion Semantics (Copestake et al., Forthcoming). The figure shown here hides some of the detail of the underspecified scope.

Such semantic ambiguity is resolved by selecting the correct derivation tree that includes the applied rules in building the tree, as shown in Figure 3. In the next phase of the Hinoki project, we are concentrating on acquiring an ontology from these semantic representations and using it to improve the parse selection (Bond et al., 2004).

3 Treebanking Using Discriminants

Selection among analyses in our set-up is done through a choice of *elementary discriminants*, basic and mostly independent contrasts between parses. These are (relatively) easy to judge by annotators. The system selects features that distinguish between different parses, and the annotator selects or rejects the features until only one parse is left. In a small number of cases, annotation may legitimately leave more than one parse active (see below). The system we used for treebanking was the [incr tsdb()] Redwoods environment[2] (Oepen et al., 2002). The number of decisions for each sentence is proportional to the log of the number of parses. The number of decisions required depends on the ambiguity of the parses and the length of the input. For Hinoki, on average, the number of decisions presented to the annotator was 27.5. However, the average number of decisions needed to disambiguate each sentence was only 2.6, plus an additional decision to accept or reject the selected parses[3]. In general, even a sentence with 100 parses requires only around 5 decisions and 1,000 parses only around 7 decisions. A graph of parse results versus number of decisions presented and required is given in Figure 6.

The primary data stored in the treebank is the derivation tree: the series of rules and lexical items the parser used to construct the parse. This, along with the grammar, can be combined to rebuild the complete HPSG sign. The annotators task is to select the appropriate derivation tree or trees. The possible derivation trees for カーテン$_2$ *kāten* "curtain" are shown in Figure 3. Nodes in the trees indicate applied rules, simplified lexical types or words. We will use it as an example to explain the annotation process. Figure 3 also displays POS tag from a separate tagger, shown in `typewriter font`.[4]

This example has two major sources of ambiguity. One is lexical: *aru* "a certain/have/be" is ambiguous between a reading as a determiner "a certain" (**det-lex**) and its use as a verb of possession "have" (**aru-verb-lex**). If it is a verb, this gives rise to further structural ambiguity in the relative clause, as discussed in Section 2. Reliable POS tags can thus resolve some ambiguity, although not all.

Overall, this five-word sentence has 6 parses. The annotator does not have to examine every tree but is instead presented with a range of 9 discriminants, as shown in Figure 4, each local to some segment of the utterance (word or phrase) and thus presenting a contrast that can be judged in isolation. Here the first column shows deduced status of discriminants (typically toggling one discriminant will rule out others), the second actual decisions, the third the discriminating rule or lexical type, the fourth the constituent spanned (with a marker showing segmentation of daughters, where it is unambiguous), and the fifth the parse trees which include the rule or lexical type.

D A	Rules / Lexical Types	Subtrees / Lexical items	Parse Trees
? ?	**rel-cl-sbj-gap**	ある物事を隠す ‖ 物	2,4,6
? ?	**rel-clause**	ある物事を隠す ‖ 物	1,3,5
- ?	**rel-cl-sbj-gap**	ある ‖ 物事	3,4
- ?	**rel-clause**	ある ‖ 物事	5,6
+ ?	**hd-specifier**	ある ‖ 物事	1,2
? ?	**subj-zpro**	隠す	2,4,6
- ?	**subj-zpro**	ある	5,6
- ?	**aru-verb-lex**	ある	3–6
+ +	**det-lex**	ある	1,2

+: positive decision
-: negative decision
?: indeterminate / unknown

Figure 4: Discriminants (marked after one is selected). *D* : deduced decisions, *A* : actual decisions

After selecting a discriminant, the system recalculates the discriminant set. Those discriminants which can be deduced to be incompatible with the decisions are marked with '−', and this information is recorded. The tool then presents to the annotator

[2]The [incr tsdb()] system, Japanese and English grammars and the Redwoods treebank of English are available from the Deep Linguistic Processing with HPSG Initiative (DELPH-IN: `http://www.delph-in.net/`).

[3]This average is over all sentences, even non-ambiguous ones, which only require a decision as to whether to accept or reject.

[4]The POS markers used in our experiment are from the ChaSen POS tag set (`http://chasen.aist-nara.ac.jp/`), we show simplified transliterated tag names.

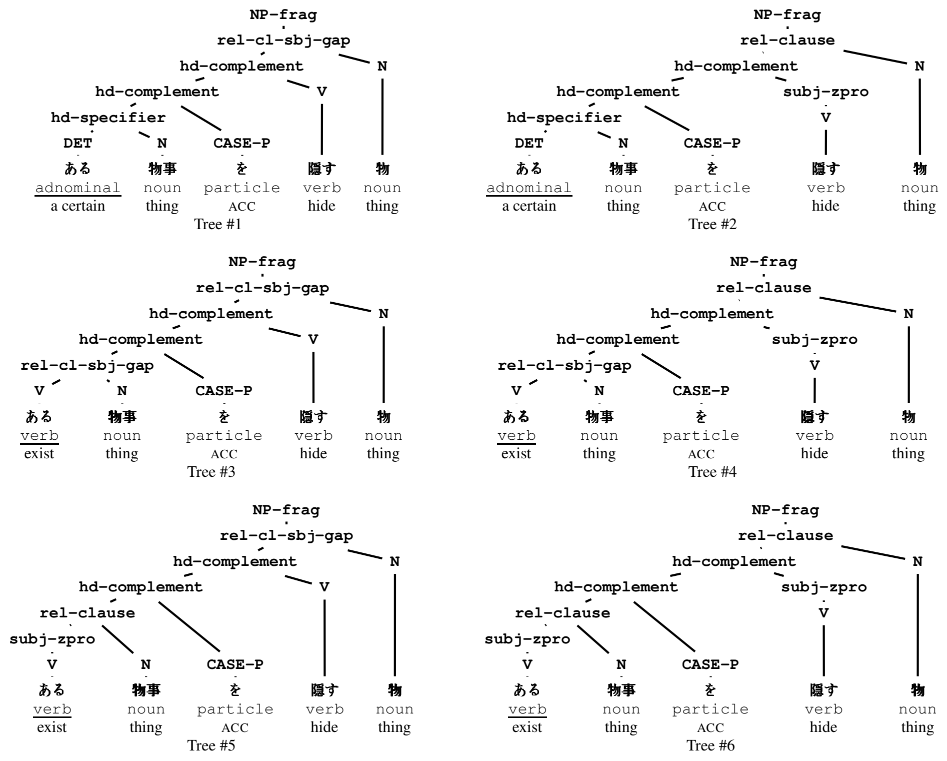


Figure 3: Derivation Trees of the Definition of カーテン$_2$ *kāten* “curtain”

only those discriminants which still select between the remaining parses, marked with ‘?’.

In this case the desired parse can be selected with a minimum of two decisions. If the first decision is that ある *aru* is a determiner (**`det-lex`**), it eliminates four parses, leaving only three discriminants (corresponding to trees #1 and #2 in Figure 3) to be decided on in the second round of decisions. Selecting 物 *mono* “thing” as the gapped subject of 隠す *kakusu* “hide” (**`rel-cl-sbj-gap`**) resolves the parse forest to the single correct derivation tree #1 in Figure 3.

The annotator also has the option of leaving some ambiguity in the treebank. For example, the verbal noun オープン *ōpun* “open” is defined with the single word 開く*aku/hiraku* “open”. This word however, has two readings: *aku* which is intransitive and *hiraku* which is transitive. As オープン *ōpun* “open” can be either transitive or intransitive, both parses are in fact correct! In such cases, the annotators were instructed to leave both parses.

Finally, the annotator has the option of rejecting all the parses presented, if none have the correct syntax and semantics. This decision has to be made even for sentences with a unique parse.

4 Using POS Tags to Blaze the Trees

Sentences in the Lexeed dictionary were already part-of-speech tagged so we investigated exploiting this information to reduce the number of decisions the annotators had to make. More generally, there are many large corpora with a subset of the information we desire already available. For example, the Kyoto Corpus (Kurohashi and Nagao, 2003) has part of speech information and dependency information, but not the detailed information available from an HPSG analysis. However, the existing information can be used to blaze[5] trees in the parse forest: that is to select or reject certain discriminants based on existing information.

Because other sources of information may not be entirely reliable, or the granularity of the information may be different from the granularity in our

[5]In forestry, to blaze is to mark a tree, usually by painting and/or cutting the bark, indicating those to be cut or the course of a boundary, road, or trail.

treebank, we felt it was important that the blazes be defeasible. The annotator can always reject the blazed decisions and retag the sentence.

In [incr tsdb()], it is currently possible to blaze using POS information. The criteria for the blazing depend on both the grammar used to make the treebank and the POS tag set. The system matches the tagged POS against the grammar's lexical hierarchy, using a one-to-many mapping of parts of speech to types of the grammar and a subsumption-based comparison. It is thus possible to write very general rules. Blazes can be positive to accept a discriminant or negative to reject it. The blaze markers are defined to be a POS tag, and then a list of lexical types and a score. The polarity of the score determines whether to accept or reject. The numerical value allows the use of a threshold, so that only those markers whose absolute value is greater than a threshold will be used. The threshold is currently set to zero: all blaze markers are used.

Due to the nature of discriminants, having two positively marked but competing discriminants for the same word will result in no trees satisfying the conditions. Therefore, it is important that only negative discriminants should be used for more general lexical types.

Hinoki uses 13 blaze markers at present, a simplified representation of them is shown in Figure 5. E.g. if ⟨verb-aux, **`v-stem-lex`**, -1.0⟩ was a blaze marker, then any sentence with a verb that has two non-auxiliary entries (e.g. *hiraku/aku* vt and vi) would be eliminated. The blaze set was derived from a conservative inspection of around 1,000 trees from an earlier round of annotation of similar data, identifying high-frequency contrasts in lexical ambiguity that can be confidently blazed from the POS granularity available for Lexeed.

POS tags	**Lexical Types in the Grammar**	**Score**
verb-aux	**`v-stem-lex`**	-1.0
verb-main	**`aspect-stem-lex`**	-1.0
noun	**`verb-stem-lex`**	-1.0
adnominal	**`noun_mod-lex-l`**	0.9
	`det-lex`	0.9
conjunction	**`n_conj-p-lex`**	0.9
	`v-coord-end-lex`	0.9
adjectival-noun	**`noun-lex`**	-1.0

Figure 5: Some Blaze Markers used in Hinoki

For the example shown in Figures 3 and 4, the blaze markers use the POS tagging of the determiner ある *aru* to mark it as **`det-lex`**. This eliminates four parses and six discriminants leaving only three to be presented to the annotator. On average, marking blazes reduced the average number of blazes presented per sentence from 27.5 to 23.8 (a reduction of 15.6%). A graphical view of number of discriminants versus parse ambiguity is shown in Figure 6.

5 Measuring Inter-Annotator Agreement

Lacking a task-oriented evaluation scenario at this point, inter-annotator agreement is our core measure of annotation consistency in Hinoki. All trees (and associated semantics) in Hinoki are derived from a computational grammar and thus should be expected to demonstrate a basic degree of internal consistency. On the other hand, the use of the grammar exposes large amounts of ambiguity to annotators that might otherwise go unnoticed. It is therefore not *a priori* clear whether the Redwoods-style approach to treebank construction as a general methodology results in a high degree of internal consistency or a comparatively low one.

	$\alpha-\beta$	$\beta-\gamma$	$\gamma-\alpha$	**Average**
Parse Agreement	63.9	68.2	64.2	65.4
Reject Agreement	4.8	3.0	4.1	4.0
Parse Disagreement	17.5	19.2	17.9	18.2
Reject Disagreement	13.7	9.5	13.8	12.4

Table 1: Exact Match Inter-annotator Agreement

Table 1 quantifies inter-annotator agreement in terms of the harshest possible measure, the proportion of sentences for which two annotators selected the exact same parse or both decided to reject all available parses. Each set was annotated by three annotators (α, β, γ). They were all native speakers of Japanese with a high score in a Japanese proficiency test (Amano and Kondo, 1998) but no linguistic training. The average annotation speed was 50 sentences an hour.

In around 19 per cent of the cases annotators chose to not fully disambiguate, keeping two or even three active parses; for these we scored $\frac{i}{j}$, with j being the number of identical pairs in the cross-product of active parses, and i the number of mismatches. One annotator keeping $\{1, 2, 3\}$, for example, and another $\{3, 4\}$ would be scored as $\frac{1}{6}$. In addition to leaving residual ambiguity, annotators opted to reject all available parses in some eight per cent of cases, usually indicating opportunities for improvement of the underlying grammar. The Parse Agreement figures (65.4%) in Table 1 are those sentences where both annotators chose one or more parses, and they showed non-zero agreement. This figure is substantially above the published figure of 52% for NeGra Brants et al. (2003). Parse Disagreement is where both chose parses, but there was no agreement. Reject Agreement shows the proportion of sentences for which both annotators found no suitable analysis. Finally Reject Disagreement is those cases were one annotator found no suitable parses, but one selected one or more.

The striking contrast between the comparatively high exact match ratios (over a random choice baseline of below seven per cent; $\kappa = 0.628$) and the low agreement between annotators on which structures to reject completely suggests that the latter type of decision requires better guidelines, ideally tests that can be operationalized.

To obtain both a more fine-grained measure and also be able to compare to related work, we computed a labeled precision f-score over derivation trees. Note that our inventory of labels is large, as they correspond in granularity to structures of the grammar: close to 1,000 lexical and 120 phrase types. As there is no 'gold' standard in contrasting two annotations, our labeled constituent measure F is the harmonic mean of standard labeled precision P (Black et al., 1991; Civit et al., 2003) applied in both 'directions': for a pair of annotators α and β, F is defined as:

$$F = \frac{2P(\alpha,\beta)P(\beta,\alpha)}{P(\alpha,\beta)+P(\beta,\alpha)}$$

As found in the discussion of exact match inter-annotator agreement over the entire treebank, there are two fundamentally distinct types of decisions made by annotators, viz. (a) elimination of unwanted ambiguity and (b) the choice of keeping at least one analysis or rejecting the entire item. Of these, only (b) applies to items that are assigned only one parse by the grammar, hence we omit unambiguous items from our labeled precision measures (a little more than twenty per cent of the total) to exclude trivial agreement from the comparison. In the same spirit, to eliminate noise hidden in pairs of items where one or both annotators opted for multiple valid parses, we further reduced the comparison set to those pairs where both annotators opted for exactly one active parse. Intersecting both conditions for pairs of annotators leaves us with subsets of around 2,500 sentences each, for which we record F values ranging from 95.1 to 97.4, see Table 2. When broken down by pairs of annotators and sets of 1,000 items each, which have been annotated in strict sequential order, F scores in Table 2 confirm that: (a) inter-annotator agreement is stable, all three annotators appear to have performed equally (well); (b) with growing experience, there is a slight increase in F scores over time, particularly when taking into account that set E exhibits a noticeably higher average ambiguity rate (1208 parses per item) than set D (820 average parses); and (c) Hinoki inter-annotator agreement compares favorably to results reported for the German NeGra (Brants, 2000) and Spanish Cast3LB (Civit et al., 2003) treebanks, both of which used manual mark-up seeded from automated POS tagging and chunking.

Compared to the 92.43 per cent labeled F score reported by Brants (2000), Hinoki achieves an 'error' (i.e. disagreement) rate of less than half, even though our structures are richer in information and should probably be contrasted with the 'edge label' F score for NeGra, which is 88.53 per cent. At the same time, it is unknown to what extent results are influenced by differences in text genre, i.e. average sentence length of our dictionary definitions is noticeably shorter than for the NeGra newspaper corpus. In addition, our measure is computed only over a subset of the corpus (those trees that can be parsed and that had multiple parses which were not rejected). If we recalculate over all 5,000 sentences, including rejected sentences (F measure of 0) and those with no ambiguity (F measure of 1) then the average F measure is 83.5, slightly worse than the score for NeGra. However, the annotation process itself identifies which the problematic sentences are, and how to improve the agreement: improve the grammar so that fewer sentences need to be rejected and then update the annotation. The Hinoki treebank is, by design, dynamic, so we expect to continue to improve the grammar and annotation continuously over the project's lifetime.

Test	$\alpha - \beta$		$\beta - \gamma$		$\gamma - \alpha$		Average
Set	#	F	#	F	#	F	F
A	507	96.03	516	96.22	481	96.24	96.19
B	505	96.79	551	96.40	511	96.57	96.58
C	489	95.82	517	95.15	477	95.42	95.46
D	454	96.83	477	96.86	447	97.40	97.06
E	480	95.15	497	96.81	484	96.57	96.51
	2435	96.32	2558	96.28	2400	96.47	96.36

Table 2: Inter-Annotator Agreement as Mutual Labeled Precision F-Score

Test	Annotator Decisions			Blazed
Set	α	β	γ	Decisions
A	2,659	2,606	3,045	416
B	2,848	2,939	2,253	451
C	1,930	2,487	2,882	468
D	2,254	2,157	2,347	397
E	1,769	2,278	1,811	412

Table 3: Number of Decisions Required

5.1 The Effects of Blazing

Table 3 shows the number of decisions per annotator, including revisions, and the number of decisions that can be done automatically by the part-of-speech blazed markers. The test sets where the annotators used the blazes are shown underlined. The final decision to accept or reject the parses was not included, as it must be made for every sentence.

The blazed test sets require far fewer annotator decisions. In order to evaluate the effect of the blazes, we compared the average number of decisions per sentence for the test sets in which some annotators used blazes and some did not (B–D). The average number of decisions went from 2.63 to 2.11, a substantial reduction of 19.5%. similarly, the time required to annotate an utterance was reduced from 83 seconds per sentence to 70, a speed up of 15.7%. We did not include A and E, as there was variation in difficulty between test sets, and it is well known that annotators improve (at least in speed of annotation) over time. Research on other projects has shown that it is normal for learning curve differences to swamp differences in tools (Wallis, 2003, p. 65). The number of decisions against the number of parses is show in Figure 6, both with and without the blazes.

6 Discussion

Annotators found the rejections the most time consuming. If a parse was eliminated, they often redid the decision process several times to be sure

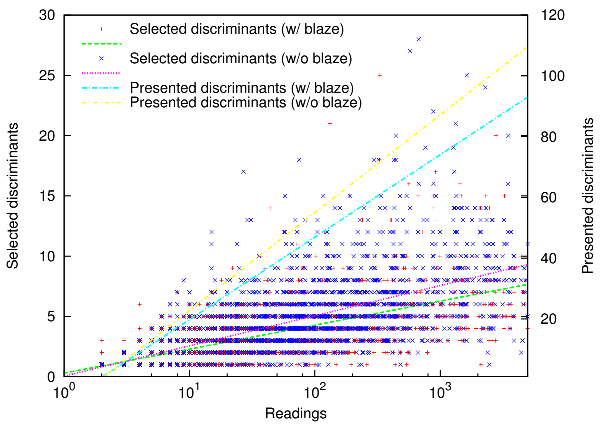


Figure 6: Number of Decisions versus Number of Parses (Test Sets B–D)

they had not eliminated the correct parse in error, which was very time consuming. This shows that the most important consideration for the success of treebanking in this manner is the quality of the grammar. Fortunately, treebanking offers direct feedback to the grammar developers. Rejected sentences identify which areas need to be improved, and because the treebank is dynamic, it can be improved when we improve the analyses in the grammar. This is a notable improvement over semi-automatically constructed grammars, such as the Penn Treebank, where many inconsistencies remain (around 4,500 types estimated by Dickinson and Meurers, 2003) and the treebank does not allow them to be identified automatically or easily updated.

Because we are simultaneously using the semantic output of the grammar in building an ontology, and the syntax and semantics are tightly coupled, the knowledge acquisition provides a further route for feedback. Extracting an ontology from the semantic representations revealed many issues with the semantics that had previously been neglected.

Our top priority for further work within Hinoki is to improve the grammar so as to both increase the cover and decrease the number of results with no acceptable parses. This will allow us to treebank a higher proportion of sentences, with even higher precision.

For more general work on treebank construction, we would like to investigate (1) using other information for blazes (syntactic constituents, dependencies, translation data) and marking blazes automatically using confident scores from existing POS taggers or parsers, (2) other agreement measures (for example agreement over the semantic representations), (3) presenting discriminants based on the semantic representations.

7 Conclusions

We conducted an experiment to measure inter-annotator agreement for the Hinoki corpus. Three annotators marked up 5,000 sentences. Sentence agreement was an unparalleled 65.4%. The method used identifies problematic annotations as a by-product, and allows the treebank to be improved as its underlying grammar improves. We also presented a method to speed up the annotation by exploiting existing part-of-speech tags. This led to a decrease in the number of annotation decisions of 19.5%.

Acknowledgments

The authors would like to thank the other members of the NTT Machine Translation Research Group, as well as Timothy Baldwin and Dan Flickinger. This research was supported by the research collaboration between the NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation and CSLI, Stanford University.

References

Anne Abeillé, editor. *Treebanks: Building and Using Parsed Corpora*. Kluwer Academic Publishers, 2003.

Shigeaki Amano and Tadahisa Kondo. Estimation of mental lexicon size with word familiarity database. In *International Conference on Spoken Language Processing*, volume 5, pages 2119–2122, 1998.

Ezra Black, Steven Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Fred Jelinek, Judith Klavans, Mark Lieberman, and Tomek Strzalkowski. A procedure for quantitatively comparing the syntactic coverage of English. In *Proceedings of the Speech and Natural Language Workshop*, pages 306–311, Pacific Grove, CA, 1991. Morgan Kaufmann.

Francis Bond, Sanae Fujita, Chikara Hashimoto, Kaname Kasahara, Shigeko Nariyama, Eric Nichols, Akira Ohtani, Takaaki Tanaka, and Shigeaki Amano. 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, 2004.

Thorsten Brants. Inter-annotator agreement for a German newspaper corpus. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*, Athens, Greece, 2000.

Thorsten Brants, Wojciech Skut, and Hans Uszkoreit. Syntactic annotation of a German newspaper corpus. In Abeillé (2003), chapter 5, pages 73–88.

David Carter. The TreeBanker: a tool for supervised training of parsed corpora. In *ACL Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, Madrid, 1997. (`http://xxx.lanl.gov/abs/cmp-lg/9705008`).

Montserrat Civit, Alicia Ageno, Borja Navarro, Núria Bufi, and Maria Antonia Martí. Qualitative and quantitative analysis of annotators' agreement in the development of Cast3LB. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, Växjö, Sweeden, 2003.

Ann Copestake, Daniel P. Flickinger, Carl Pollard, and Ivan A. Sag. Minimal Recursion Semantics. An introduction. *Journal of Research in Language and Computation*, Forthcoming.

Markus Dickinson and W. Detmar Meurers. Detecting inconsistencies in treebanks. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, Växjö, Sweeden, 2003.

Kaname Kasahara, Hiroshi Sato, Francis Bond, Takaaki Tanaka, Sanae Fujita, Tomoko Kanasugi, and Shigeaki Amano. Construction of a Japanese semantic lexicon: Lexeed. SIG NLC-159, IPSJ, Tokyo, 2004. (in Japanese).

Adam Kilgariff and Joseph Rosenzweig. Framework and results for English SENSEVAL. *Computers and the Humanities*, 34 (1–2):15–48, 2000. Special Issue on SENSEVAL.

Sadao Kurohashi and Makoto Nagao. Building a Japanese parsed corpus — while improving the parsing system. In Abeillé (2003), chapter 14, pages 249–260.

Stephan Oepen, Dan Flickinger, and Francis Bond. Towards holistic grammar engineering and testing — grafting treebank maintenance into the grammar revision cycle. In *Beyond Shallow Analyses — Formalisms and Statistical Modeling for Deep Analysis (Workshop at IJCNLP-2004)*, Hainan Island, 2004. (`http://www-tsujii.is.s.u-tokyo.ac.jp/bsa/`).

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christoper D. Manning, Dan Flickinger, and Thorsten Brant. The LinGO redwoods treebank: Motivation and preliminary applications. In *19th International Conference on Computational Linguistics: COLING-2002*, pages 1253–7, Taipei, Taiwan, 2002.

Carl Pollard and Ivan A. Sag. *Head Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, 1994.

Melanie Siegel and Emily M. Bender. 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*, Taipei, 2002.

Sean Wallis. Completing parsed corpora: From correction to evolution. In Abeillé (2003), chapter 4, pages 61–71.

A Dynamic Bayesian Framework to Model Context and Memory in Edit Distance Learning: An Application to Pronunciation Classification

Karim Filali and **Jeff Bilmes**[*]
Departments of Computer Science & Engineering and Electrical Engineering
University of Washington
Seattle, WA 98195, USA
{karim@cs,bilmes@ee}.washington.edu

Abstract

Sitting at the intersection between statistics and machine learning, Dynamic Bayesian Networks have been applied with much success in many domains, such as speech recognition, vision, and computational biology. While Natural Language Processing increasingly relies on statistical methods, we think they have yet to use Graphical Models to their full potential. In this paper, we report on experiments in learning edit distance costs using Dynamic Bayesian Networks and present results on a pronunciation classification task. By exploiting the ability within the DBN framework to rapidly explore a large model space, we obtain a 40% reduction in error rate compared to a previous transducer-based method of learning edit distance.

1 Introduction

Edit distance (ED) is a common measure of the similarity between two strings. It has a wide range of applications in classification, natural language processing, computational biology, and many other fields. It has been extended in various ways; for example, to handle simple (Lowrance and Wagner, 1975) or (constrained) block transpositions (Leusch et al., 2003), and other types of block operations (Shapira and Storer, 2003); and to measure similarity between graphs (Myers et al., 2000; Klein, 1998) or automata (Mohri, 2002).

Another important development has been the use of data-driven methods for the automatic learning of edit costs, such as in (Ristad and Yianilos, 1998) in the case of string edit distance and in (Neuhaus and Bunke, 2004) for graph edit distance.

In this paper we revisit the problem of learning string edit distance costs within the Graphical Models framework. We apply our method to a pronunciation classification task and show significant improvements over the standard Levenshtein distance (Levenshtein, 1966) and a previous transducer-based learning algorithm.

In section 2, we review a stochastic extension of the classic string edit distance. We present our DBN-based edit distance models in section 3 and show results on a pronunciation classification task in section 4. In section 5, we discuss the computational aspects of using our models. We end with our conclusions and future work in section 6.

2 Stochastic Models of Edit Distance

Let $s_1^m = s_1 s_2 ... s_m$ be a *source* string over a source alphabet $\mathcal{A}$, and m the length of the string. s_i^j is the substring $s_i ... s_j$ and s_i^j is equal to the empty string, ϵ, when $i > j$. Likewise, t_1^n denotes a *target* string over a target alphabet $\mathcal{B}$, and n the length of t_1^n.

A source string can be transformed into a target string through a sequence of *edit operations*. We write $\langle s, t \rangle$ ($(s, t) \neq (\epsilon, \epsilon)$) to denote an *edit operation* in which the symbol s is replaced by t. If $s = \epsilon$ and $t \neq \epsilon$, $\langle s, t \rangle$ is an *insertion*. If $s \neq \epsilon$ and $t = \epsilon$, $\langle s, t \rangle$ is a *deletion*. When $s \neq \epsilon$, $t \neq \epsilon$ and $s \neq t$, $\langle s, t \rangle$ is a *substitution*. In all other cases, $\langle s, t \rangle$ is an *identity*.

The string edit distance, $d(s_1^m, t_1^n)$ between s_1^m and t_1^n is defined as the minimum weighted sum of

[*]This material was supported by NSF under Grant No. ISS-0326276.

the number of deletions, insertions, and substitutions required to transform s_1^m into t_1^n (Wagner and Fischer, 1974). A $O(m \cdot n)$ Dynamic Programming (DP) algorithm exists to compute the ED between two strings. The algorithm is based on the following recursion:

$$d(s_1^i, t_1^j) = min \left(\begin{array}{l} d(s_1^{i-1}, t_1^j) + \gamma(\langle s_i, \epsilon \rangle), \\ d(s_1^i, t_1^{j-1}) + \gamma(\langle \epsilon, t_j \rangle), \\ d(s_1^{i-1}, t_1^{j-1}) + \gamma(\langle s_i, t_j \rangle) \end{array} \right)$$

with $d(\epsilon, \epsilon)=0$ and $\gamma : \{\langle s, t\rangle | (s, t) \neq (\epsilon, \epsilon)\} \rightarrow \Re_+$ a cost function. When γ maps non-identity edit operations to unity and identities to zero, string ED is often referred to as the *Levenshtein distance*.

To learn the edit distance costs from data, Ristad and Yianilos (1998) use a generative model (henceforth referred to as the *RY model*) based on a memoryless transducer of string pairs. Below we summarize their main idea and introduce our notation, which will be useful later on.

We are interested in modeling the joint probability $P(S_1^m{=}s_1^m, T_1^n{=}t_1^n \,|\, \theta)$ of observing the source/target string pair (s_1^m, t_1^n) given model parameters θ. S_i (resp. T_i), $1{\le}i{\le}m$, is a random variable (RV) associated with the event of observing a source (resp. target) symbol at position i.[1]

To model the edit operations, we introduce a hidden RV, Z, that takes values in $(\mathcal{A} \cup \epsilon \times \mathcal{B} \cup \epsilon) \setminus \{(\epsilon, \epsilon)\}$. Z can be thought of as a *random vector* with two components, $Z^{(s)}$ and $Z^{(t)}$.

We can then write the joint probability $P(s_1^m, t_1^n \,|\, \theta)$ as

$$P(s_1^m, t_1^n \,|\, \theta) = \sum_{\{z_1^\ell : v(z_1^\ell) = <s_1^m, t_1^n>, \, max(m,n) \le \ell \le m+n\}} \sum P(Z_1^\ell{=}z_1^\ell, s_1^m, t_1^n \,|\, \theta) \quad (1)$$

where $v(z_1^\ell)$ is the *yield* of the sequence z_1^ℓ: the string pair output by the transducer.

Equation 1 says that the probability of a particular pair of strings is equal to the sum of the probabilities of all possible ways to generate the pair by concatenating the edit operations $z_1 ... z_\ell$. If we make the assumption that there is no dependence between edit operations, we call our model *memoryless*. $P(Z_1^\ell, s_1^m, t_1^n \,|\, \theta)$ can then be factored as $\Pi_i P(Z_i, s_1^m, t_1^n \,|\, \theta)$. In addition, we call the model *context-independent* if we can write $Q(z_i) = P(Z_i{=}z_i, s_1^m, t_1^n \,|\, \theta)$, $1{<}i{<}\ell$, where $z_i{=}\langle z_i^{(s)}, z_i^{(t)} \rangle$, in the form

$$Q(z_i) \propto \begin{cases} f^{ins}(t_{b_i}) & \text{for } z_i^{(s)} = \epsilon; z_i^{(t)} = t_{b_i} \\ f^{del}(s_{a_i}) & \text{for } z_i^{(s)} = s_{a_i}; z_i^{(t)} = \epsilon \\ f^{sub}(s_{a_i}, t_{b_i}) & \text{for } (z_i^{(s)}, z_i^{(t)}) = (s_{a_i}, t_{b_i}) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $\sum_z Q(z) = 1$; $a_i = \sum_{j=1}^{i-1} 1_{\{z_j^{(s)} \neq \epsilon\}}$ (resp. b_i) is the index of the source (resp. target) string generated up to the ith edit operation; and f^{ins},f^{del},and f^{sub} are functions mapping to $[0, 1]$.[2] Context independence is not to be taken here to mean Z_i does not depend on s_{a_i} or t_{b_i}. It depends on them through the *global context* which forces Z_1^ℓ to generate (s_1^m, t_1^n). The RY model is *memoryless and context-independent* (MCI).

Equation 2, also implicitly enforces the *consistency constraint* that the pair of symbols output, $(z_i^{(s)}, z_i^{(t)})$, agrees with the actual pair of symbols, (s_{a_i}, t_{b_i}), that needs to be generated at step i in order for the total yield, $v(z_1^\ell)$, to equal the string pair.

The RY stochastic model is similar to the one introduced earlier by Bahl and Jelinek (1975). The difference is that the Bahl model is memoryless and *context-dependent* (MCD); the f functions are now indexed by s_{a_i} (or t_{a_i}, or both) such that $\sum_z Q_{s_{a_i}}(z) = 1 \; \forall s_{a_i}$. In general, context dependence can be extended to include up to the whole source (and/or target) string, $s_1^{a_i - 1}, s_{a_i}, s_{a_i+1}^m$. Several other types of dependence can be exploited as will be discussed in section 3.

Both the Ristad and the Bahl transducer models give exponentially smaller probability to longer strings and edit sequences. Ristad presents an alternate explicit model of the joint probability of the length of the source and target strings. In this parametrization the probability of the length of an edit sequence does not necessarily decrease geometrically. A similar effect can be achieved by modeling the length of the hidden edit sequence explicitly (see section 3).

3 DBNs for Learning Edit Distance

Dynamic Bayesian Networks (DBNs), of which Hidden Markov Models (HMMs) are the most fa-

[1] We follow the convention of using capital letters for random variables and lowercase letters for instantiations of random variables.

[2] By convention, $s_{a_i} = \epsilon$ for $a_i > m$. Likewise, $t_{b_i} = \epsilon$ if $b_i > n$. $f^{ins}(\epsilon) = f^{del}(\epsilon) = f^{sub}(\epsilon, \epsilon) = 0$. This takes care of the case when we are past the end of a string.

mous representative, are well suited for modeling stochastic temporal processes such as speech and neural signals. DBNs belong to the larger family of Graphical Models (GMs). In this paper, we restrict ourselves to the class of DBNs and use the terms DBN and GM interchangeably. For an example in which Markov Random Fields are used to compute a context-sensitive edit distance see (Wei, 2004).[3]

There is a large body of literature on DBNs and algorithms associated with them. To briefly define a graphical model, it is a way of representing a (*factored*) probability distribution using a graph. Nodes of the graph correspond to random variables; and edges to dependence relations between the variables.[4] To do *inference* or parameter learning using DBNs, various generic exact or approximate algorithms exist (Lauritzen, 1996; Murphy, 2002; Bilmes and Bartels, 2003). In this section we start by introducing a graphical model for the MCI transducer then present four additional classes of DBN models: context-dependent, memory (where an edit operation can depend on past operations), direct (HMM-like), and length models (in which we explicitly model the length of the sequence of edits to avoid the exponential decrease in likelihood of longer sequences). A few other models are discussed in section 4.2.

3.1 Memoryless Context-independent Model

Fig. 1 shows a DBN representation of the memoryless context-independent transducer model (section 2). The graph represents a *template* which consists, in general, of three parts: a *prologue*, a *chunk*, and an *epilogue*. The chunk is repeated as many times as necessary to model sequences of arbitrary length. The product of *unrolling* the template is a Bayesian Network organized into a given number of *frames*. The prologue and the epilogue often differ from the chunk because they model boundary conditions, such as ensuring that the end of both strings is reached at or before the last frame.

Associated with each node is a probability function that maps the node's parent values to the values the node can take. We will refer to that function as a

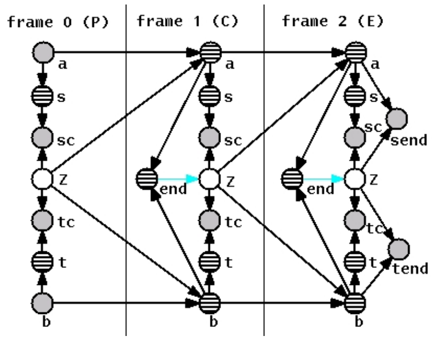


Figure 1: *DBN for the memory-less transducer model. Unshaded nodes are hidden nodes with probabilistic dependencies with respect to their parents. Nodes with stripes are deterministic hidden nodes, i.e., they take a unique value for each configuration of their parents. Filled nodes are observed (they can be either stochastic or deterministic). The graph template is divided into three frames. The center frame is repeated $m+n-2$ times to yield a graph with a total of $m+n$ frames, the maximum number of edit operations needed to transform s_1^m into t_1^n. Outgoing light edges mean the parent is a switching variable with respect to the child: depending on the value of the switching RV, the child uses different CPTs and/or a different parent set.*

conditional probability table (CPT).

Common to all the frames in fig. 1, are position RVs, a and b, which encode the current positions in the source and target strings resp.; source and target symbols, s and t; the hidden edit operation, Z; and consistency nodes sc and tc, which enforce the consistency constraint discussed in section 2. Because of symmetry we will explain the upper half of the graph involving the source string unless the target half is different. We drop subscripts when the frame number is clear from the context.

In the first frame, a and b are observed to have value 1, the first position in both strings. a and b determine the value of the symbols s and t. Z takes a random value $\langle z^{(s)}, z^{(t)} \rangle$. sc has the fixed observed value 1. The only configurations of its parents, Z and s, that satisfy $P(sc=1|s,z) > 0$ are such that ($Z^{(s)} = s$) or ($Z^{(s)} = \epsilon$ and $Z \neq \langle \epsilon, \epsilon \rangle$). This is the consistency constraint in equation 2.

In the following frame, the position RV a_2 depends on a_1 and Z_1. If Z_1 is an insertion (i.e. $Z_1^{(s)} = \epsilon$: the source symbol in the first frame is

[3]While the *Markov Edit Distance* introduced in the paper takes local statistical dependencies into account, the edit costs are still fixed and not corpus-driven.

[4]The concept of *d-separation* is useful to read independence relations encoded by the graph (Lauritzen, 1996).

not output), then a_2 retains the same value as a_1; otherwise a_2 is incremented by 1 to point to the next symbol in the source string.

The end RV is an indicator of when we are past the end of both source and target strings ($a > m$ and $b > n$). end is also a *switching parent* of Z; when $end = 0$, the CPT of Z is the same as described above: a distribution over edit operations. When $end = 1$, Z takes, with probability 1, a fixed value outside the range of edit operations but consistent with s and t. This ensures 1) no "null" state ($\langle\epsilon, \epsilon\rangle$) is required to fill in the value of Z until the end of the graph is reached; our likelihoods and model parameters therefore do not become dependent on the amount of "null" padding; and 2) no probability mass is taken from the other states of Z as is the case with the special termination symbol # in the original RY model. We found empirically that the use of either a null or an end state hurts performance to a small but significant degree.

In the last frame, two new nodes make their appearance. $send$ and $tend$ ensure we are *at or past* the end of the two strings (the RV end only checks that we are past the end). That is why $send$ depends on both a and Z. If $a > m$, $send$ (observed to be 1) is 1 with probability 1. If $a < m$, then $P(send{=}1) = 0$ and the whole sequence Z_1^ℓ has zero probability. If $a = m$, then $send$ only gets probability greater than zero if Z is not an insertion. This ensures the last source symbol is indeed consumed.

Note that we can obtain the equivalent of the total edit distance cost by using *Viterbi inference* and adding a $cost_i$ variable as a deterministic child of the random variable Z_i : in each frame the cost is equal to $cost_{i-1}$ plus 0 when Z_i is an identity, or plus 1 otherwise.

3.2 Context-dependent Model

Adding context dependence in the DBN framework is quite natural. In fig. 2, we add edges from s_i, $sprev_i$, and $snext_i$ to Z_i. The sc node is no longer required because we can enforce the consistency constraint via the CPT of Z given its parents. $snext_i$ is an RV whose value is set to the symbol at the $a_i{+}1$ position of the string, i.e., $snext_i{=}s_{a_i+1}$. Likewise, $sprev_i = s_{a_i-1}$. The Bahl model (1975) uses a dependency on s_i only. Note that s_{i-1} is not necessarily equal to s_{a_i-1}. Conditioning on s_{i-1} induces an

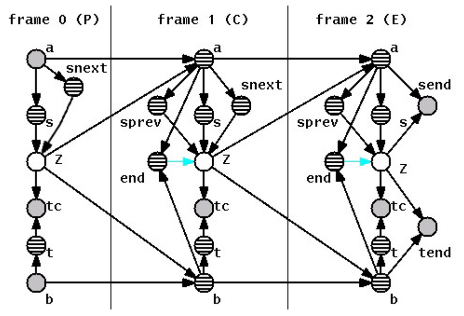


Figure 2: *Context-dependent model.*

indirect dependence on whether there was an insertion in the previous step because $s_{i-1} = s_i$ might be correlated with the event $Z_{i-1}^{(s)} = \epsilon$.

3.3 Memory Model

Memory models are another easy extension of the basic model as fig. 3 shows. Depending on whether the variable H_{i-1} linking Z_{i-1} to Z_i is stochastic or deterministic, there are several models that can be implemented; for example, a latent factor memory model when H is stochastic. The cardinality of H determines how much the information from one frame to the other is "summarized." With a deterministic implementation, we can, for example, specify the usual $P(Z_i|Z_{i-1})$ memory model when H is a simple copy of Z or have Z_i depend on the type of edit operation in the previous frame.

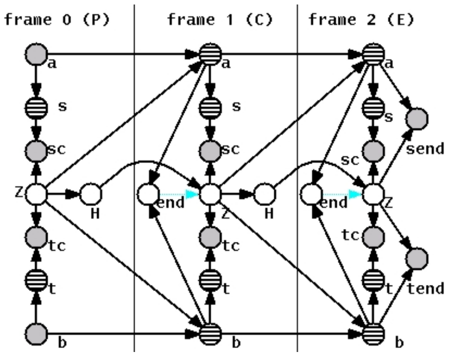


Figure 3: *Memory model. Depending on the type of dependency between Z_i and H_i, the model can be latent variable based or it can implement a deterministic dependency on a function of Z_i*

3.4 Direct Model

The *direct model* in fig. 4 is patterned on the classic HMM, where the unrolled length of graph is the same as the length of the sequence of observations. The key feature of this model is that we are required to consume a target symbol per frame. To achieve that, we introduce two RVs, ins, with cardinality 2, and del, with cardinality at most m. The dependency of del on ins is to ensure the two events never happen concomitantly. At each frame, a is incremented either by the value of del in the case of a (possibly block) deletion or by zero or one depending on whether there was an insertion in the previous frame. An insertion also forces s to take value ϵ.

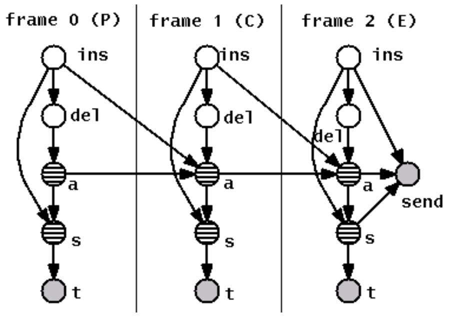


Figure 4: *Direct model.*

In essence the direct model is not very different from the context-dependent model in that here too we learn the conditional probabilities $P(t_i|s_i)$ (which are implicit in the CD model).

3.5 Length Model

While this model (fig. 5) is more complex than the previous ones, much of the network structure is "control logic" necessary to simulate variable length-unrolling of the graph template. The key idea is that we have a new stochastic hidden RV, $inclen$, whose value added to that of the RV $inilen$ determines the number of edit operations we are allowed. A counter variable, $counter$ is used to keep track of the frame number and when the required number is reached, the RV $atReqLen$ is triggered. If at that point we have just reached the end of one of the strings while the end of the other one is reached in this frame or a previous one, then the variable end is *explained* (it has positive probability). Otherwise, the entire sequence of edit operations up to that point has zero probability.

4 Pronunciation Classification

In pronunciation classification we are given a *lexicon*, which consists of words and their corresponding *canonical pronunciations*. We are also provided with *surface pronunciations* and asked to find the most likely corresponding words. Formally, for each

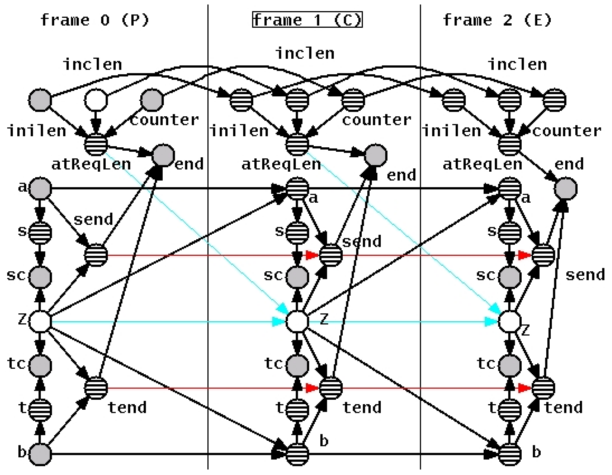


Figure 5: *Length unrolling model.*

surface form, t_1^n, we need to find the set of words $\hat{W}$ s.t. $\hat{W} = argmax_w P(w|t_1^n)$. There are several ways we could model the probability $P(w|t_1^n)$. One way is to assume a generative model whereby a word w and a surface pronunciation t_1^n are related via an underlying canonical pronunciation s_1^m of w and a stochastic process that explains the transformation from s_1^m to t_1^n. This is summarized in equation 3. $C(w)$ denotes the set of canonical pronunciations of w.

$$\hat{W} = \underset{w}{\text{argmax}} \sum_{s_1^m \in C(w)} P(w|s_1^m) P(s_1^m, t_1^n) \quad (3)$$

If we assume uniform probabilities $P(w|s_1^m)$ ($s_1^m \in C(w)$) and use the max approximation in place of the sum in eq. 3 our classification rule becomes

$$\hat{W} = \{w | \hat{S} \cap C(w) \neq \emptyset, \hat{S} = \underset{s_1^m}{\text{argmax}}\ P(s_1^m, t_1^n)\} \quad (4)$$

It is straightforward to create a DBN to model the joint probability $P(w, s_1^m, t_1^n)$ by adding a word RV and a canonical pronunciation RV on top of any of the previous models.

There are other pronunciation classification approaches with various emphases. For example, Rentzepopoulos and Kokkinakis (1996) use HMMs to convert phoneme sequences to their most likely orthographic forms in the absence of a lexicon.

4.1 Data

We use Switchboard data (Godfrey et al., 1992) that has been hand annotated in the context of the Speech Transcription Project (STP) described in (Greenberg et al., 1996). Switchboard consists of spontaneous informal conversations recorded over the phone. Because of the informal non-scripted nature of the speech and the variety of speakers, the corpus presents much variety in word pronunciations, which can significantly deviate from the prototypical pronunciations found in a lexicon. Another source of pronunciation variability is the noise introduced during the annotation of speech segments. Even when the phone labels are mostly accurate, the start and end time information is not as precise and it affects how boundary phones get aligned to the word sequence. As a reference pronunciation dictionary we use a lexicon of the 2002 Switchboard speech recognition evaluation. The lexicon contains 40000 entries, but we report results on a reduced dictionary[5] with 5000 entries corresponding to only those words that appear in our train and test sets. Ristad and Yianilos use a few additional lexicons, some of which are corpus-derived. We did reproduce their results on the different types of lexicons.

For testing we randomly divided STP data into 9495 training words (corresponding to 9545 pronunciations) and 912 test words (901 pronunciations). For the Levenshtein and MCI results only, we performed ten-fold cross validation to verify we did not pick a non-representative test set. Our models are implemented using GMTK, a general-purpose DBN tool originally created to explore different speech recognition models (Bilmes and Zweig, 2002). As a sanity check, we also implemented the MCI model in C following RY's algorithm.

The error rate is computed by calculating, for each pronunciation form, the fraction of words that are correctly hypothesized and averaging over the test set. For example if the classifier returns five words for a given pronunciation, and two of the words are correct, the error rate is 3/5*100%.

Three EM iterations are used for training. Additional iterations overtrained our models.

4.2 Results

Table 1 summarizes our results using DBN based models. The basic MCI model does marginally better than the Levenshtein edit distance. This is consistent with the finding in RY: their gains come from the joint learning of the probabilities $P(w|s_1^m)$ and $P(s_1^m, t_1^n)$. Specifically, the word model accounts for much of their gains over the Levenshtein distance. We use uniform priors and the simple classification rule in eq. 4. We feel it is more compelling that we are able to significantly improve upon standard edit distance and the MCI model without using any lexicon or word model.

Memory Models Performance improves with the addition of a direct dependence of Z_i on Z_{i-1}. The biggest improvement (27.65% ER) however comes from conditioning on $Z_{i-1}^{(t)}$, the target symbol that is hypothesized in the previous step. There was no gain when conditioning on the type of edit operation in the previous frame.

Context Models Interestingly, the exact opposite from the memory models is happening here when we condition on the source context (versus conditioning on the target context). Conditioning on s_i gets us to 21.70%. With s_i, s_{i-1} we can further reduce the error rate to 20.26%. However, when we add a third dependency, the error rate worsens to 29.32%, which indicates a number of parameters too high for the given amount of training data. Backoff, interpolation, or state clustering might all be appropriate strategies here.

Position Models Because in the previous models, when conditioning on the past, boundary conditions dictate that we use a different CPT in the first frame, it is fair to wonder whether part of the gain we witness is due to the implicit dependence on the source-target string position. The (small) improvement due to conditioning on b_i indicates there is such dependence. Also, the fact that the target position is more informative than the source one is likely due to the misalignments we observed in the phonetically transcribed corpus, whereby the first or last phones would incorrectly be aligned with the previous or next word resp. I.e., the model might be learning to not put much faith in the start and end positions of the target string, and thus it boosts deletion and insertion probabilities at those positions. We have also conditioned on coarser-grained positions (beginning, middle, and end of string) but obtained the same results as with the fine-grained dependency.

Length Models Modeling length helps to a small extent when it is added to the MCI and MCD models. Belying the assumption motivating this model, we found that the distribution over the RV $inclen$ (which controls how much the edit sequence extends

[5] Equivalent to the *E2* lexicon in RY.

beyond the length of the source string) is skewed towards small values of $inclen$. This indicates on that insertions are rare when the source string is longer than the target one and vice-versa for deletions.

Direct Model The low error rate obtained by this model reflects its similarity to the context-dependent model. From the two sets of results, it is clear that source string context plays a crucial role in predicting canonical pronunciations from corpus ones. We would expect additional gains from modeling context dependencies across time here as well.

Model	Z_i Dependencies	% Err rate
Lev	none	**35.97**
Baseline	none	**35.55**
Memory	Z_{i-1}	30.05
	editOperationType(Z_{i-1})	36.16
	stochastic binary H_{i-1}	33.87
	$Z^{(s)}_{i-1}$	29.62
	$Z^{(t)}_{i-1}$	**27.65**
Context	s_i	21.70
	t_i	32.06
	s_i, s_{i-1}	**20.26**
	t_i, t_{i-1}	28.21
	s_i, s_{i-1}, s_{a_i+1}	29.32
	s_i, s_{a_i+1} (s_{a_i-1} in last frame)	23.14
	s_i, s_{a_i-1} (s_{a_i+1} in first frame)	23.15
Position	a_i	33.80
	b_i	**31.06**
	a_i, b_i	34.17
Mixed	b_i,s_i	**22.22**
	$Z^{(t)}_{i-1}$,s_i	24.26
Length	none	33.56
	s_i	**20.03**
Direct	none	**23.70**

Table 1: *DBN based model results summary.*

When we combine the best position-dependent or memory models with the context-dependent one, the error rate decreases (from 31.31% to 25.25% when conditioning on b_i and s_i; and from 28.28% to 25.75% when conditioning on $z^{(t)}_{i-1}$ and s_i) but not to the extent conditioning on s_i alone decreases error rate. Not shown in table 1, we also tried several other models, which although they are able to produce reasonable alignments (in the sense that the Levenshtein distance would result in similar alignments) between two given strings, they have extremely poor discriminative ability and result in error rates higher than 90%. One such example is a model in which Z_i depends on both s_i and t_i. It is easy to see where the problem lies with this model once one considers that two very different strings might still get a higher likelihood than more similar pair because, given s and t s.t. $s \neq t$, the probability of identity is obviously zero and that of insertion or deletion can be quite high; and when $s = t$, the probability of insertion (or deletion) is still positive. We observe the same non-discriminative behavior when we replace, in the MCI model, Z_i with a hidden RV X_i, where X_i takes as values one of the four edit operations.

5 Computational Considerations

The computational complexity of inference in a graphical model is related to the state space of the largest clique (maximal complete subgraph) in the graph. In general, finding the smallest such clique is NP-complete (Arnborg et al., 1987).

In the case of the MCI model, however, it is not difficult to show that the smallest such clique contains all the RVs within a frame and the complexity of doing inference is order $O(mn \cdot max(m, n))$. The reason there is a complexity gap is that the source and target position variables are indexed by the frame number and we do not exploit the fact that even though we arrive at a given source-target position pair along different edit sequence paths at different frames, the position pair is really the same regardless of its frame index. We are investigating generic ways of exploiting this constraint.

In practice, however, state space pruning can significantly reduce the running time of DBN inference. Ukkonen (1985) reduces the complexity of the classic edit distance to $O(d \cdot max(m, n))$, where d is the edit distance. The intuition there is that, assuming a small edit distance, the most likely alignments are such that the source position does not diverge too much from the target position. The same intuition holds in our case: if the source and the target position do not get too far out of sync, then at each step, only a small fraction of the $m \cdot n$ possible source-target position configurations need be considered.

The direct model, for example, is quite fast in practice because we can restrict the cardinality of the *del* RV to a constant c (i.e. we disallow long-span deletions, which for certain applications is a reasonable restriction) and make inference linear in n with a running time constant proportional to c^2.

6 Conclusion

We have shown how the problem of learning edit distance costs from data can be modeled quite naturally using Dynamic Bayesian Networks even though the problem lacks the temporal or order constraints that other problems such as speech recognition exhibit. This gives us confidence that other important problems such as machine translation can benefit from a Graphical Models perspective. Machine translation presents a fresh set of challenges because of the large combinatorial space of possible alignments between the source string and the target.

There are several extensions to this work that we intend to implement or have already obtained preliminary results on. One is simple and block transposition. Another natural extension is modeling edit distance of multiple strings.

It is also evident from the large number of dependency structures that were explored that our learning algorithm would benefit from a structure learning procedure. Maximum likelihood optimization might, however, not be appropriate in this case, as exemplified by the failure of some models to discriminate between different pronunciations. Discriminative methods have been used with significant success in training HMMs. Edit distance learning could benefit from similar methods.

References

S. Arnborg, D. G. Corneil, and A. Proskurowski. 1987. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284.

L. R. Bahl and F. Jelinek. 1975. Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition. *Trans. on Information Theory*, 21:404–411.

J. Bilmes and C. Bartels. 2003. On triangulating dynamic graphical models. In *Uncertainty in Artificial Intelligence: Proceedings of the 19th Conference*, pages 47–56. Morgan Kaufmann.

J. Bilmes and G. Zweig. 2002. The Graphical Models Toolkit: An open source software system for speech and time-series processing. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *ICASSP*, volume 1, pages 517–520.

S. Greenberg, J. Hollenback, and D. Ellis. 1996. Insights into spoken language gleaned from phonetic transcription of the switchboard corpus. In *ICSLP*, pages S24–27.

P. N. Klein. 1998. Computing the edit-distance between unrooted ordered trees. In *Proceedings of 6th Annual European Symposium*, number 1461, pages 91–102.

S.L. Lauritzen. 1996. *Graphical Models*. Oxford Science Publications.

G. Leusch, N. Ueffing, and H. Ney. 2003. A novel string-to-string distance measure with applications to machine translation evaluation. In *Machine Translation Summit IX*, pages 240–247.

V. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.*, 10:707–710.

R. Lowrance and R. A. Wagner. 1975. An extension to the string-to-string correction problem. *J. ACM*, 22(2):177–183.

M. Mohri. 2002. Edit-distance of weighted automata. In *CIAA*, volume 2608 of *Lecture Notes in Computer Science*, pages 1–23. Springer.

K. Murphy. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, U.C. Berkeley, Dept. of EECS, CS Division.

R. Myers, R.C. Wison, and E.R. Hancock. 2000. Bayesian graph edit distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22:628–635.

M. Neuhaus and H. Bunke. 2004. A probabilistic approach to learning costs for graph edit distance. In *ICPR*, volume 3, pages 389–393.

P. A. Rentzepopoulos and G. K. Kokkinakis. 1996. Efficient multilingual phoneme-to-grapheme conversion based on hmm. *Comput. Linguist.*, 22(3):351–376.

E. S. Ristad and P. N. Yianilos. 1998. Learning string edit distance. *Trans. on Pattern Recognition and Machine Intelligence*, 20(5):522–532.

D. Shapira and J. A. Storer. 2003. Large edit distance with multiple block operations. In *SPIRE*, volume 2857 of *Lecture Notes in Computer Science*, pages 369–377. Springer.

E. Ukkonen. 1985. Algorithms for approximate string matching. *Inf. Control*, 64(1-3):100–118.

R. A. Wagner and M. J. Fischer. 1974. The string-to-string correction problem. *J. ACM*, 21(1):168–173.

J. Wei. 2004. Markov edit distance. *Trans. on Pattern Analysis and Machine Intelligence*, 26(3):311–321.

Learning Stochastic OT Grammars: A Bayesian approach using Data Augmentation and Gibbs Sampling

Ying Lin[*]
Department of Linguistics
University of California, Los Angeles
Los Angeles, CA 90095
yinglin@ucla.edu

Abstract

Stochastic Optimality Theory (Boersma, 1997) is a widely-used model in linguistics that did not have a theoretically sound learning method previously. In this paper, a Markov chain Monte-Carlo method is proposed for learning Stochastic OT Grammars. Following a Bayesian framework, the goal is finding the posterior distribution of the grammar given the relative frequencies of input-output pairs. The Data Augmentation algorithm allows one to simulate a joint posterior distribution by iterating two conditional sampling steps. This Gibbs sampler constructs a Markov chain that converges to the joint distribution, and the target posterior can be derived as its marginal distribution.

1 Introduction

Optimality Theory (Prince and Smolensky, 1993) is a linguistic theory that dominates the field of phonology, and some areas of morphology and syntax. The standard version of OT contains the following assumptions:

- A grammar is a set of ordered constraints ($\{C_i : i = 1, \cdots, N\}, >$);
- Each constraint C_i is a function: $\Sigma^* \rightarrow \{0, 1, \cdots\}$, where Σ^* is the set of strings in the language;
- Each underlying form u corresponds to a set of candidates $GEN(u)$. To obtain the unique surface form, the candidate set is successively filtered according to the order of constraints, so that only the most harmonic candidates remain after each filtering. If only 1 candidate is left in the candidate set, it is chosen as the optimal output.

The popularity of OT is partly due to learning algorithms that induce constraint ranking from data. However, most of such algorithms cannot be applied to noisy learning data. Stochastic Optimality Theory (Boersma, 1997) is a variant of Optimality Theory that tries to quantitatively predict linguistic variation. As a popular model among linguists that are more engaged with empirical data than with formalisms, Stochastic OT has been used in a large body of linguistics literature.

In Stochastic OT, constraints are regarded as independent normal distributions with unknown means and fixed variance. As a result, the stochastic constraint hierarchy generates systematic linguistic variation. For example, consider a grammar with 3 constraints, $C_1 \sim N(\mu_1, \sigma^2)$, $C_2 \sim N(\mu_2, \sigma^2)$, $C_3 \sim N(\mu_3, \sigma^2)$, and 2 competing candidates for a given input x:

	$p(.)$	C_1	C_2	C_3
$x \sim y_1$	.77	0	0	1
$x \sim y_2$	.23	1	1	0

Table 1: A Stochastic OT grammar with 1 input and 2 outputs

[*]The author thanks Bruce Hayes, Ed Stabler, Yingnian Wu, Colin Wilson, and anonymous reviewers for their comments.

The probabilities $p(.)$ are obtained by repeatedly sampling the 3 normal distributions, generating the winning candidate according to the ordering of constraints, and counting the relative frequencies in the outcome. As a result, the grammar will assign non-zero probabilities to a given set of outputs, as shown above.

The learning problem of Stochastic OT involves fitting a grammar $G \in R^N$ to a set of candidates with frequency counts in a corpus. For example, if the learning data is the above table, we need to find an estimate of $G = (\mu_1, \mu_2, \mu_3)$[1] so that the following ordering relations hold with certain probabilities:

$$\begin{array}{l} \max\{C_1, C_2\} > C_3; \textit{ with probability } .77 \\ \max\{C_1, C_2\} < C_3; \textit{ with probability } .23 \end{array} \quad (1)$$

The current method for fitting Stochastic OT models, used by many linguists, is the Gradual Learning Algorithm (GLA) (Boersma and Hayes, 2001). GLA looks for the correct ranking values by using the following heuristic, which resembles gradient descent. First, an input-output pair is sampled from the data; second, an ordering of the constraints is sampled from the grammar and used to generate an output; and finally, the means of the constraints are updated so as to minimize the error. The updating is done by adding or subtracting a "plasticity" value that goes to zero over time. The intuition behind GLA is that it does "frequency matching", i.e. looking for a better match between the output frequencies of the grammar and those in the data.

As it turns out, GLA does not work in all cases[2], and its lack of formal foundations has been questioned by a number of researchers (Keller and Asudeh, 2002; Goldwater and Johnson, 2003). However, considering the broad range of linguistic data that has been analyzed with Stochastic OT, it seems unadvisable to reject this model because of the absence of theoretically sound learning methods. Rather, a general solution is needed to evaluate Stochastic OT as a model for linguistic variation. In this paper, I introduce an algorithm for learning Stochastic OT grammars using Markov chain Monte-Carlo methods. Within a Bayesian framework, the learning problem is formalized as finding the *posterior distribution* of ranking values (G) given the information on constraint interaction based on input-output pairs (D). The posterior contains all the information needed for linguists' use: for example, if there is a grammar that will generate the exact frequencies as in the data, such a grammar will appear as a mode of the posterior.

In computation, the posterior distribution is simulated with MCMC methods because the likelihood function has a complex form, thus making a maximum-likelihood approach hard to perform. Such problems are avoided by using the *Data Augmentation* algorithm (Tanner and Wong, 1987) to make computation feasible: to simulate the posterior distribution $G \sim p(G|D)$, we augment the parameter space and simulate a joint distribution $(G, Y) \sim p(G, Y|D)$. It turns out that by setting Y as the value of constraints that observe the desired ordering, simulating from $p(G, Y|D)$ can be achieved with a *Gibbs sampler*, which constructs a Markov chain that converges to the joint posterior distribution (Geman and Geman, 1984; Gelfand and Smith, 1990). I will also discuss some issues related to efficiency in implementation.

2 The difficulty of a maximum-likelihood approach

Naturally, one may consider "frequency matching" as estimating the grammar based on the maximum-likelihood criterion. Given a set of constraints and candidates, the data may be compiled in the form of (1), on which the likelihood calculation is based. As an example, given the grammar and data set in Table 1, the likelihood of d="$\max\{C1, C2\} > C3$" can be written as $P(d|\mu_1, \mu_2, \mu_3)$=

$$1 - \int_{-\infty}^{0} \int_{-\infty}^{0} \frac{1}{2\pi\sigma^2} \exp\left\{ -\frac{\vec{f}_{xy} \cdot \Sigma \cdot \vec{f}_{xy}^T}{2} \right\} dx\, dy$$

where $\vec{f}_{xy} = (x - \mu_1 + \mu_3, y - \mu_2 + \mu_3)$, and Σ is the identity covariance matrix. The integral sign follows from the fact that both $C_1 - C_2$, $C_2 - C_3$ are normal, since each constraint is independently normally distributed.

If we treat each data as independently generated by the grammar, then the likelihood will be a product of such integrals (multiple integrals if many constraints are interacting). One may attempt to maximize such a likelihood function using numerical

[1]Up to translation by an additive constant.

[2]Two examples included in the experiment section. See 6.3.

methods[3], yet it appears to be desirable to avoid likelihood calculations altogether.

3 The missing data scheme for learning Stochastic OT grammars

The Bayesian approach tries to explore $p(G|D)$, the posterior distribution. Notice if we take the usual approach by using the relationship $p(G|D) \propto p(D|G) \cdot p(G)$, we will encounter the same problem as in Section 2. Therefore we need a feasible way of sampling $p(G|D)$ without having to derive the closed-form of $p(D|G)$.

The key idea here is the so-called "missing data" scheme in Bayesian statistics: in a complex model-fitting problem, the computation can sometimes be greatly simplified if we treat part of the unknown parameters as data and fit the model in successive stages. To apply this idea, one needs to observe that Stochastic OT grammars are learned from *ordinal data*, as seen in (1). In other words, only one aspect of the structure generated by those normal distributions — the ordering of constraints — is used to generate outputs.

This observation points to the possibility of treating the sample values of constraints $\vec{y} = (y_1, y_2, \cdots, y_N)$ that satisfy the ordering relations as missing data. It is appropriate to refer to them as "missing" because a language learner obviously cannot observe real numbers from the constraints, which are postulated by linguistic theory. When the observed data are augmented with missing data and become a *complete data* model, computation becomes significantly simpler. This type of idea is officially known as *Data Augmentation* (Tanner and Wong, 1987). More specifically, we also make the following intuitive observations:

- The complete data model consists of 3 random variables: the observed ordering relations D, the grammar G, and the missing samples of constraint values Y that generate the ordering D.
- G and Y are interdependent:
  - For each fixed d, values of Y that respect d can be obtained easily once G is given: we just sample from $p(Y|G)$ and only keep those that observe d. Then we let d vary with its frequency in the data, and obtain a sample of $p(Y|G, D)$;
  - Once we have the values of Y that respect the ranking relations D, G becomes independent of D. Thus, sampling G from $p(G|Y, D)$ becomes the same as sampling from $p(G|Y)$.

4 Gibbs sampler for the joint posterior — $p(G, Y|D)$

The interdependence of G and Y helps design iterative algorithms for sampling $p(G, Y|D)$. In this case, since each step samples from a conditional distribution ($p(G|Y, D)$ or $p(Y|G, D)$), they can be combined to form a Gibbs sampler (Geman and Geman, 1984). In the same order as described in Section 3, the two conditional sampling steps are implemented as follows:

1. Sample an ordering relation d according to the prior $p(D)$, which is simply normalized frequency counts; sample a vector of constraint values $y = \{y_1, \cdots, y_N\}$ from the normal distributions $N(\mu_1^{(t)}, \sigma^2), \cdots, N(\mu_N^{(t)}, \sigma^2)$ such that y observes the ordering in d;

2. Repeat Step 1 and obtain M samples of missing data: $y^1, \cdots, y^M$; sample $\mu_i^{(t+1)}$ from $N(\sum_j y_i^j/M, \sigma^2/M)$.

The grammar $G = (\mu_1, \cdots, \mu_N)$, and the superscript $^{(t)}$ represents a sample of G in iteration t. As explained in 3, Step 1 samples missing data from $p(Y|G, D)$, and Step 2 is equivalent to sampling from $p(G|Y, D)$, by the conditional independence of G and D given Y. The normal posterior distribution $N(\sum_j y_i^j/M, \sigma^2/M)$ is derived by using $p(G|Y) \propto p(Y|G)p(G)$, where $p(Y|G)$ is normal, and $p(G) \sim N(\mu_0, \sigma_0)$ is chosen to be an non-informative prior with $\sigma_0 \to \infty$.

M (the number of missing data) is not a crucial parameter. In our experiments, M is set to the total number of observed forms[4]. Although it may seem that σ^2/M is small for a large M and does not play

[3]Notice even computing the gradient is non-trivial.

[4]Other choices of M, e.g. $M = 1$, lead to more or less the same running time.

a significant role in the sampling of $\mu_i^{(t+1)}$, the variance of the sampling distribution is a necessary ingredient of the Gibbs sampler[5].

Under fairly general conditions (Geman and Geman, 1984), the Gibbs sampler iterates these two steps until it converges to a unique stationary distribution. In practice, convergence can be monitored by calculating cross-sample statistics from multiple Markov chains with different starting points (Gelman and Rubin, 1992). After the simulation is stopped at convergence, we will have obtained a perfect sample of $p(G, Y|D)$. These samples can be used to derive our target distribution $p(G|D)$ by simply keeping all the G components, since $p(G|D)$ is a marginal distribution of $p(G, Y|D)$. Thus, the sampling-based approach gives us the advantage of doing inference without performing any integration.

5 Computational issues in implementation

In this section, I will sketch some key steps in the implementation of the Gibbs sampler. Particular attention is paid to sampling $p(Y|G, D)$, since a direct implementation may require an unrealistic running time.

5.1 Computing $p(D)$ from linguistic data

The prior probability $p(D)$ determines the number of samples (missing data) that are drawn under each ordering relation. The following example illustrates how the ordering D and $p(D)$ are calculated from data collected in a linguistic analysis. Consider a data set that contains 2 inputs and a few outputs, each associated with an observed frequency in the lexicon:

		C1	C2	C3	C4	C5	Freq.
x_1	y_{11}	0	1	0	1	0	4
	y_{12}	1	0	0	0	0	3
	y_{13}	0	1	1	0	1	0
	y_{14}	0	0	1	0	0	0
x_2	y_{21}	1	1	0	0	0	3
	y_{22}	0	0	1	1	1	0

Table 2: A Stochastic OT grammar with 2 inputs

The three ordering relations (corresponding to 3 attested outputs) and $p(D)$ are computed as follows:

Ordering Relation D	$p(D)$
$\left\{\begin{array}{r} C1>\max\{C2,C4\} \\ \max\{C3,C5\}>C4 \\ C3>\max\{C2,C4\}\end{array}\right.$	.4
$\left\{\begin{array}{r} \max\{C2,C4\}>C1 \\ \max\{C2,C3,C5\}>C1 \\ C3>C1\end{array}\right.$	.3
$\max\{C3,C4,C5\} > \max\{C1,C2\}$	.3

Table 3: The ordering relations D and $p(D)$ computed from Table 2.

Here each ordering relation has several conjuncts, and the number of conjuncts is equal to the number of competing candidates for each given input. These conjuncts need to hold simultaneously because each winning candidate needs to be more harmonic than all other competing candidates. The probabilities $p(D)$ are obtained by normalizing the frequencies of the surface forms in the original data. This will have the consequence of placing more weight on lexical items that occur frequently in the corpus.

5.2 Sampling $p(Y|G, D)$ under complex ordering relations

A direct implementation $p(Y|G, d)$ is straightforward: 1) first obtain N samples from N Gaussian distributions; 2) check each conjunct to see if the ordering relation is satisfied. If so, then keep the sample; if not, discard the sample and try again.

However, this can be highly inefficient in many cases. For example, if m constraints appear in the ordering relation d and the sample is rejected, the $N - m$ random numbers for constraints not appearing in d are also discarded. When d has several conjuncts, the chance of rejecting samples for irrelevant constraints is even greater.

In order to save the generated random numbers, the vector Y can be decomposed into its 1-dimensional components $(Y_1, Y_2, \cdots, Y_N)$. The problem then becomes sampling $p(Y_1, \cdots, Y_N|G, D)$. Again, we may use conditional sampling to draw y_i one at a time: we keep $y_{j\neq i}$ and d fixed[6], and draw y_i so that d holds for y. There are now two cases: if d holds regardless of y_i, then any sample from $N(\mu_i^{(t)}, \sigma^2)$ will do; otherwise, we will need to draw y_i from a truncated

[5]As required by the proof in (Geman and Geman, 1984).

[6]Here we use $y_{j\neq i}$ for all components of y except the i-th dimension.

normal distribution.

To illustrate this idea, consider an example used earlier where d="$\max\{c_1, c_2\} > c_3$", and the initial sample and parameters are $(y_1^{(0)}, y_2^{(0)}, y_3^{(0)}) = (\mu_1^{(0)}, \mu_2^{(0)}, \mu_3^{(0)}) = (1, -1, 0)$.

Sampling dist.	Y_1	Y_2	Y_3
$p(Y_1 \mid \mu_1, Y_1 > y_3)$	$\underline{2.3799}$	-1.0000	0
$p(Y_2 \mid \mu_2)$	2.3799	$\underline{-0.7591}$	0
$p(Y_3 \mid \mu_3, Y_3 < y_1)$	2.3799	-0.7591	$\underline{-1.0328}$
$p(Y_1 \mid \mu_1)$	$\underline{-1.4823}$	-0.7591	-1.0328
$p(Y_2 \mid \mu_2, Y_2 > y_3)$	-1.4823	$\underline{2.1772}$	-1.0328
$p(Y_3 \mid \mu_3, Y_3 < y_2)$	-1.4823	2.1772	$\underline{1.0107}$

Table 4: Conditional sampling steps for $p(Y|G, d) = p(Y_1, Y_2, Y_3|\mu_1, \mu_2, \mu_3, d)$

Notice that in each step, the sampling density is either just a normal, or a truncated normal distribution. This is because we only need to make sure that d will continue to hold for the next sample $y^{(t+1)}$, which differs from $y^{(t)}$ by just 1 constraint.

In our experiment, sampling from truncated normal distributions is realized by using the idea of *rejection sampling*: to sample from a truncated normal[7] $\pi_c(x) = \frac{1}{Z(c)} \cdot N(\mu, \sigma) \cdot I_{\{x>c\}}$, we first find an *envelope* density function $g(x)$ that is easy to sample directly, such that $\pi_c(x)$ is uniformly bounded by $M \cdot g(x)$ for some constant M that does not depend on x. It can be shown that once each sample x from $g(x)$ is rejected with probability $r(x) = 1 - \frac{\pi_c(x)}{M \cdot g(x)}$, the resulting histogram will provide a perfect sample for $\pi_c(x)$. In the current work, the exponential distribution $g(x) = \lambda \exp\{-\lambda x\}$ is used as the envelope, with the following choices for λ and the rejection ratio $r(x)$, which have been optimized to lower the rejection rate:

$$\begin{aligned} \lambda &= \frac{c + \sqrt{c + 4\sigma^2}}{2\sigma^2} \\ r(x) &= \exp\left\{\frac{(x+c)^2}{2} + \lambda_0(x+c) - \frac{\sigma^2\lambda_0^2}{2}\right\} \end{aligned}$$

Putting these ideas together, the final version of Gibbs sampler is constructed by implementing Step 1 in Section 4 as a sequence of conditional sampling steps for $p(Y_i|Y_{j\neq i}, d)$, and combining them with the sampling of $p(G|Y, D)$. Notice the order in which Y_i is updated is fixed, which makes our implementation an instance of the *systematic-scan* Gibbs sampler (Liu, 2001). This implementation may be improved even further by utilizing the structure of the ordering relation d, and optimizing the order in which Y_i is updated.

5.3 Model identifiability

Identifiability is related to the uniqueness of solution in model fitting. Given N constraints, a grammar $G \in R^N$ is not identifiable because $G + C$ will have the same behavior as G for any constant $C = (c_0, \cdots, c_0)$. To remove translation invariance, in Step 2 the average ranking value is subtracted from G, such that $\sum_i \mu_i = 0$.

Another problem related to identifiability arises when the data contains the so-called "categorical domination", i.e., there may be data of the following form:

$c_1 > c_2$ *with probability* 1.

In theory, the mode of the posterior tends to infinity and the Gibbs sampler will not converge. Since having categorical dominance relations is a common practice in linguistics, we avoid this problem by truncating the posterior distribution[8] by $I_{|\mu|<K}$, where K is chosen to be a positive number large enough to ensure that the model be identifiable. The role of truncation/renormalization may be seen as a strong prior that makes the model identifiable on a bounded set.

A third problem related to identifiability occurs when the posterior has multiple modes, which suggests that multiple grammars may generate the same output frequencies. This situation is common when the grammar contains interactions between many constraints, and greedy algorithms like GLA tend to find one of the many solutions. In this case, one can either introduce extra ordering relations or use informative priors to sample $p(G|Y)$, so that the inference on the posterior can be done with a relatively small number of samples.

5.4 Posterior inference

Once the Gibbs sampler has converged to its stationary distribution, we can use the samples to make var-

[7]Notice the truncated distribution needs to be re-normalized in order to be a proper density.

[8]The implementation of sampling from truncated normals is the same as described in 5.2.

ious inferences on the posterior. In the experiments reported in this paper, we are primarily interested in the mode of the posterior marginal[9] $p(\mu_i|D)$, where $i = 1, \cdots, N$. In cases where the posterior marginal is symmetric and uni-modal, its mode can be estimated by the sample median.

In real linguistic applications, the posterior marginal may be a skewed distribution, and many modes may appear in the histogram. In these cases, more sophisticated non-parametric methods, such as kernel density estimation, can be used to estimate the modes. To reduce the computation in identifying multiple modes, a mixture approximation (by EM algorithm or its relatives) may be necessary.

6 Experiments

6.1 Ilokano reduplication

The following Ilokano grammar and data set, used in (Boersma and Hayes, 2001), illustrate a complex type of constraint interaction: the interaction between the three constraints: *COMPLEX-ONSET, ALIGN, and $IDENT_{BR}$([long]) cannot be factored into interactions between 2 constraints. For any given candidate to be optimal, the constraint that prefers such a candidate must simultaneously dominate the other two constraints. Hence it is not immediately clear whether there is a grammar that will assign equal probability to the 3 candidates.

/HRED-bwaja/	$p(.)$	*C-ONS	AL	I_{BR}
bu:.bwa.ja	.33	1	0	1
bwaj.bwa.ja	.33	2	0	0
bub.wa.ja	.33	0	1	0

Table 5: Data for Ilokano reduplication.

Since it does not address the problem of identifiability, the GLA does not always converge on this data set, and the returned grammar does not always fit the input frequencies exactly, depending on the choice of parameters[10].

In comparison, the Gibbs sampler converges quickly[11], regardless of the parameters. The result suggests the existence of a unique grammar that will assign equal probabilities to the 3 candidates. The posterior samples and histograms are displayed in Figure 1. Using the median of the marginal posteriors, the estimated grammar generates an exact fit to the frequencies in the input data.

[9]Note $G = (\mu_1, \cdots, \mu_N)$, and $p(\mu_i|D)$ is a marginal of $p(G|D)$.

[10]B &H reported results of averaging many runs of the algorithm. Yet there appears to be significant randomness in each run of the algorithm.

[11]Within 1000 iterations.

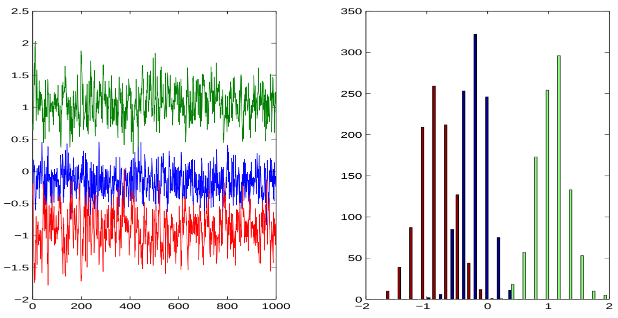


Figure 1: Posterior marginal samples and histograms for Experiment 2.

6.2 Spanish diminutive suffixation

The second experiment uses linguistic data on Spanish diminutives and the analysis proposed in (Arbisi-Kelm, 2002). There are 3 base forms, each associated with 2 diminutive suffixes. The grammar consists of 4 constraints: ALIGN(TE,Word,R), MAX-OO(V), DEP-IO and BaseTooLittle. The data presents the problem of learning from noise, since no Stochastic OT grammar can provide an exact fit to the data: the candidate [ubita] violates an extra constraint compared to [liri.ito], and [ubasita] violates the same constraint as [liryosito]. Yet unlike [lityosito], [ubasita] is not observed.

Input	Output	Freq.	**A**	**M**	**D**	**B**
/uba/	[ubita]	10	0	1	0	1
	[ubasita]	0	1	0	0	0
/mar/	[marEsito]	5	0	0	1	0
	[marsito]	5	0	0	0	1
/liryo/	[liri.ito]	9	0	1	0	0
	[liryosito]	1	1	0	0	0

Table 6: Data for Spanish diminutive suffixation.

In the results found by GLA, [marEsito] always has a lower frequency than [marsito] (See Table 7). This is not accidental. Instead it reveals a problematic use of heuristics in GLA[12]: since the constraint **B** is violated by [ubita], it is always demoted whenever the underlying form /uba/ is encountered during learning. Therefore, even though the expected

[12]Thanks to Bruce Hayes for pointing out this problem.

model assigns equal values to μ_3 and μ_4 (corresponding to **D** and **B**, respectively), μ_3 is always less than μ_4, simply because there is more chance of penalizing **D** rather than **B**. This problem arises precisely because of the heuristic (i.e. demoting the constraint that prefers the wrong candidate) that GLA uses to find the target grammar.

The Gibbs sampler, on the other hand, does not depend on heuristic rules in its search. Since modes of the posterior $p(\mu_3|D)$ and $p(\mu_4|D)$ reside in negative infinity, the posterior is truncated by $I_{\mu_i<K}$, with $K = 6$, based on the discussion in 5.3. Results of the Gibbs sampler and two runs of GLA[13] are reported in Table 7.

Input	Output	Obs	Gibbs	GLA$_1$	GLA$_2$
/uba/	[ubita]	100%	95%	96%	96%
	[ubasita]	0%	5%	4%	4%
/mar/	[marEsito]	50%	50%	38%	45%
	[marsito]	50%	50%	62%	55%
/liryo/	[liri.ito]	90%	95%	96%	91.4%
	[liryosito]	10%	5%	4%	8.6%

Table 7: Comparison of Gibbs sampler and GLA

7 A comparison with Max-Ent models

Previously, problems with the GLA[14] have inspired other OT-like models of linguistic variation. One such proposal suggests using the more well-known *Maximum Entropy* model (Goldwater and Johnson, 2003). In Max-Ent models, a grammar G is also parameterized by a real vector of weights $w = (w_1, \cdots, w_N)$, but the conditional likelihood of an output y given an input x is given by:

$$p(y|x) = \frac{\exp\{\sum_i w_i f_i(y,x)\}}{\sum_z \exp\{\sum_i w_i f_i(z,x)\}} \qquad (2)$$

where $f_i(y, x)$ is the violation each constraint assigns to the input-output pair (x, y).

Clearly, Max-Ent is a rather different type of model from Stochastic OT, not only in the use of constraint ordering, but also in the objective function (conditional likelihood rather than likelihood/posterior). However, it may be of interest to compare these two types of models. Using the same data as in 6.2, results of fitting Max-Ent (using conjugate gradient descent) and Stochastic OT (using Gibbs sampler) are reported in Table 8:

Input	Output	Obs	SOT	ME	ME$_{sm}$
/uba/	[ubita]	100%	95%	100%	97.5%
	[ubasita]	0%	5%	0%	2.5%
/mar/	[marEsito]	50%	50%	50%	48.8%
	[marsito]	50%	50%	50%	51.2%
/liryo/	[liri.ito]	90%	95%	90%	91.4%
	[liryosito]	10%	5%	10%	8.6%

Table 8: Comparison of Max-Ent and Stochastic OT models

It can be seen that the Max-Ent model, in the absence of a smoothing prior, fits the data perfectly by assigning positive weights to constraints **B** and **D**. A less exact fit (denoted by ME$_{sm}$) is obtained when the smoothing Gaussian prior is used with $\mu_i = 0$, $\sigma_i^2 = 1$. But as observed in 6.2, an exact fit is impossible to obtain using Stochastic OT, due to the difference in the way variation is generated by the models. Thus it may be seen that Max-Ent is a more powerful class of models than Stochastic OT, though it is not clear how the Max-Ent model's descriptive power is related to generative linguistic theories like phonology.

Although the abundance of well-behaved optimization algorithms has been pointed out in favor of Max-Ent models, it is the author's hope that the MCMC approach also gives Stochastic OT a similar underpinning. However, complex Stochastic OT models often bring worries about identifiability, whereas the convexity property of Max-Ent may be viewed as an advantage[15].

8 Discussion

From a non-Bayesian perspective, the MCMC-based approach can be seen as a randomized strategy for learning a grammar. Computing resources make it possible to explore the entire space of grammars and discover where good hypotheses are likely to occur. In this paper, we have focused on the frequently visited areas of the hypothesis space.

It is worth pointing out that the Graduate Learning Algorithm can also be seen from this perspective. An examination of the GLA shows that when the plasticity term is fixed, parameters found by GLA also form a Markov chain $G^{(t)} \in R^N, t = 1, 2, \cdots$. Therefore, assuming the model is identifiable, it

[13]The two runs here both use 0.002 and 0.0001 as the final plasticity. The initial plasticity and the iterations are set to 2 and 1.0e7. Slightly better fits can be found by tuning these parameters, but the observation remains the same.

[14]See (Keller and Asudeh, 2002) for a summary.

[15]Concerns about identifiability appear much more frequently in statistics than in linguistics.

seems possible to use GLA in the same way as the MCMC methods: rather than forcing it to stop, we can run GLA until it reaches stationary distribution, if it exists.

However, it is difficult to interpret the results found by this "random walk-GLA" approach: the stationary distribution of GLA may not be the target distribution — the posterior $p(G|D)$. To construct a Markov chain that converges to $p(G|D)$, one may consider turning GLA into a real MCMC algorithm by designing *reversible jumps*, or the *Metropolis algorithm*. But this may not be easy, due to the difficulty in likelihood evaluation (including likelihood ratio) discussed in Section 2.

In contrast, our algorithm provides a general solution to the problem of learning Stochastic OT grammars. Instead of looking for a Markov chain in R^N, we go to a higher dimensional space $R^N \times R^N$, using the idea of data augmentation. By taking advantage of the interdependence of G and Y, the Gibbs sampler provides a Markov chain that converges to $p(G, Y|D)$, which allows us to return to the original subspace and derive $p(G|D)$ — the target distribution. Interestingly, by adding more parameters, the computation becomes simpler.

9 Future work

This work can be extended in two directions. First, it would be interesting to consider other types of OT grammars, in connection with the linguistics literature. For example, the variances of the normal distribution are fixed in the current paper, but they may also be treated as unknown parameters (Nagy and Reynolds, 1997). Moreover, constraints may be parameterized as mixture distributions, which represent other approaches to using OT for modeling linguistic variation (Anttila, 1997).

The second direction is to introduce informative priors motivated by linguistic theories. It is found through experimentation that for more sophisticated grammars, identifiability often becomes an issue: some constraints may have multiple modes in their posterior marginal, and it is difficult to extract modes in high dimensions[16]. Therefore, use of priors is needed in order to make more reliable inferences. In addition, priors also have a linguistic appeal, since current research on the "initial bias" in language acquisition can be formulated as priors (e.g. *Faithfulness Low* (Hayes, 2004)) from a Bayesian perspective.

Implementing these extensions will merely involve modifying $p(G|Y, D)$, which we leave for future work.

[16]Notice that posterior marginals do not provide enough information for modes of the joint distribution.

References

Anttila, A. (1997). *Variation in Finnish Phonology and Morphology*. PhD thesis, Stanford University.

Arbisi-Kelm, T. (2002). An analysis of variability in Spanish diminutive formation. Master's thesis, UCLA, Los Angeles.

Boersma, P. (1997). How we learn variation, optionality, probability. In *Proceedings of the Institute of Phonetic Sciences 21*, pages 43–58, Amsterdam. University of Amsterdam.

Boersma, P. and Hayes, B. P. (2001). Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry*, 32:45–86.

Gelfand, A. and Smith, A. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410).

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7:457–472.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Goldwater, S. and Johnson, M. (2003). Learning OT constraint rankings using a Maximum Entropy model. In Spenader, J., editor, *Proceedings of the Workshop on Variation within Optimality Theory*, Stockholm.

Hayes, B. P. (2004). Phonological acquisition in optimality theory: The early stages. In Kager, R., Pater, J., and Zonneveld, W., editors, *Fixing Priorities: Constraints in Phonological Acquisition*. Cambridge University Press.

Keller, F. and Asudeh, A. (2002). Probabilistic learning algorithms and Optimality Theory. *Linguistic Inquiry*, 33(2):225–244.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*. Number 33 in Springer Statistics Series. Springer-Verlag, Berlin.

Nagy, N. and Reynolds, B. (1997). Optimality theory and variable word-final deletion in Faetar. *Language Variation and Change*, 9.

Prince, A. and Smolensky, P. (1993). *Optimality Theory: Constraint Interaction in Generative Grammar*. Forthcoming.

Tanner, M. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398).

Contrastive Estimation: Training Log-Linear Models on Unlabeled Data[*]

Noah A. Smith and **Jason Eisner**
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{nasmith,jason}@cs.jhu.edu

Abstract

Conditional random fields (Lafferty et al., 2001) are quite effective at sequence labeling tasks like shallow parsing (Sha and Pereira, 2003) and named-entity extraction (McCallum and Li, 2003). CRFs are *log-linear*, allowing the incorporation of arbitrary features into the model. To train on *unlabeled* data, we require *unsupervised* estimation methods for log-linear models; few exist. We describe a novel approach, *contrastive estimation*. We show that the new technique can be intuitively understood as exploiting *implicit negative evidence* and is computationally efficient. Applied to a sequence labeling problem—POS tagging given a tagging dictionary and unlabeled text—contrastive estimation outperforms EM (with the same feature set), is more robust to degradations of the dictionary, and can largely recover by modeling additional features.

1 Introduction

Finding linguistic structure in raw text is not easy. The classical forward-backward and inside-outside algorithms try to guide probabilistic models to discover structure in text, but they tend to get stuck in local maxima (Charniak, 1993). Even when they avoid local maxima (e.g., through clever initialization) they typically deviate from human ideas of what the "right" structure is (Merialdo, 1994).

One strategy is to incorporate domain knowledge into the model's structure. Instead of blind HMMs or PCFGs, one could use models whose features are crafted to pay attention to a range of domain-specific linguistic cues. *Log-linear* models can be so crafted and have already achieved excellent performance when trained on *annotated* data, where they are known as "maximum entropy" models (Ratnaparkhi et al., 1994; Rosenfeld, 1994).

Our goal is to learn log-linear models from *unannotated* data. Since the forward-backward and inside-outside algorithms are instances of Expectation-Maximization (EM) (Dempster et al., 1977), a natural approach is to construct EM algorithms that handle log-linear models. Riezler (1999) did so, then resorted to an approximation because the true objective function was hard to normalize.

Stepping back from EM, we may generally envision parameter estimation for probabilistic modeling as pushing probability mass toward the training examples. We must consider not only where the learner pushes the mass, but also *from where* the mass is *taken*. In this paper, we describe an alternative to EM: *contrastive estimation* (CE), which (unlike EM) explicitly states the source of the probability mass that is to be given to an example.[1]

One reason is to make normalization *efficient*. Indeed, CE generalizes EM and other practical techniques used to train log-linear models, including conditional estimation (for the supervised case) and Riezler's approximation (for the unsupervised case).

The other reason to use CE is to improve *accuracy*. CE offers an additional way to inject domain knowledge into unsupervised learning (Smith and Eisner, 2005). CE hypothesizes that each positive example in training implies a domain-specific set of examples which are (for the most part) degraded (§2). This class of *implicit negative evidence* provides the source of probability mass for the observed example. We discuss the application of CE to log-linear models in §3.

[*]This work was supported by a Fannie and John Hertz Foundation fellowship to the first author and NSF ITR grant IIS-0313193 to the second author. The views expressed are not necessarily endorsed by the sponsors. The authors also thank three anonymous ACL reviewers for helpful comments, colleagues at JHU CLSP (especially David Smith and Roy Tromble) and Miles Osborne for insightful feedback, and Eric Goldlust and Markus Dreyer for Dyna language support.

[1]Not to be confused with *contrastive divergence* minimization (Hinton, 2003), a technique for training products of experts.

We are particularly interested in log-linear models over *sequences*, like the conditional random fields (CRFs) of Lafferty et al. (2001) and weighted CFGs (Miyao and Tsujii, 2002). For a given sequence, implicit negative evidence can be represented as a *lattice* derived by finite-state operations (§4). Effectiveness of the approach on POS tagging using unlabeled data is demonstrated (§5). We discuss future work (§6) and conclude (§7).

2 Implicit Negative Evidence

Natural language is a delicate thing. For any plausible sentence, there are many slight perturbations of it that will make it implausible. Consider, for example, the first sentence of this section. Suppose we choose one of its six words at random and remove it; on this example, odds are two to one that the resulting sentence will be ungrammatical. Or, we could randomly choose two adjacent words and transpose them; none of the results are valid conversational English. The learner we describe here takes into account not only the observed positive example, but also a set of similar but deprecated negative examples.

2.1 Learning setting

Let $\vec{x} = \langle x_1, x_2, ...\rangle$, be our observed example sentences, where each $x_i \in \mathcal{X}$, and let $y_i^* \in \mathcal{Y}$ be the unobserved correct hidden structure for x_i (e.g., a POS sequence). We seek a model, parameterized by $\vec{\theta}$, such that the (unknown) correct analysis y_i^* is the best analysis for x_i (under the model). If y_i^* were observed, a variety of training criteria would be available (see Tab. 1), but y_i^* is unknown, so none apply. Typically one turns to the EM algorithm (Dempster et al., 1977), which locally maximizes

$$\prod_i p\left(X = x_i \mid \vec{\theta}\right) = \prod_i \sum_{y \in \mathcal{Y}} p\left(X = x_i, Y = y \mid \vec{\theta}\right) \quad (1)$$

where X is a random variable over sentences and Y a random variable over analyses (notation is often abbreviated, eliminating the random variables). An often-used alternative to EM is a class of so-called Viterbi approximations, which iteratively find the probabilistically-best $\hat{y}$ and then, on each iteration, solve a supervised problem (see Tab. 1).

joint likelihood (JL)	$\prod_i p\left(x_i, y_i^* \mid \vec{\theta}\right)$
conditional likelihood (CL)	$\prod_i p\left(y_i^* \mid x_i, \vec{\theta}\right)$
classification accuracy (Juang and Katagiri, 1992)	$\sum_i \delta(y_i^*, \hat{y}(x_i))$
expected classification accuracy (Klein and Manning, 2002)	$\sum_i p\left(y_i^* \mid x_i, \vec{\theta}\right)$
negated boosting loss (Collins, 2000)	$-\sum_i p\left(y_i^* \mid x_i, \vec{\theta}\right)^{-1}$
margin (Crammer and Singer, 2001)	γ s.t. $\lVert\vec{\theta}\rVert \leq 1; \forall i, \forall y \neq y_i^*,$ $\vec{\theta} \cdot (\vec{f}(x_i, y_i^*) - \vec{f}(x_i, y)) \geq \gamma$
expected local accuracy (Altun et al., 2003)	$\prod_i \prod_j p\left(\ell_j(Y) = \ell_j(y_i^*) \mid x_i, \vec{\theta}\right)$

Table 1: Various supervised training criteria. All functions are written so as to be maximized. None of these criteria are available for *unsupervised* estimation because they all depend on the correct label, y^*.

2.2 A new approach: contrastive estimation

Our approach instead maximizes

$$\prod_i p\left(X_i = x_i \mid X_i \in \mathcal{N}(x_i), \vec{\theta}\right) \quad (2)$$

where the "neighborhood" $\mathcal{N}(x_i) \subseteq \mathcal{X}$ is a set of implicit negative examples plus the example x_i itself. As in EM, $p(x_i \mid ..., \vec{\theta})$ is found by marginalizing over hidden variables (Eq. 1). Note that the $x' \in \mathcal{N}(x_i)$ are not treated as hard negative examples; we merely seek to move probability mass from them to the observed x.

The neighborhood of x, $\mathcal{N}(x)$, contains examples that are perturbations of x. We refer to the mapping $\mathcal{N} : \mathcal{X} \to 2^{\mathcal{X}}$ as the neighborhood function, and the optimization of Eq. 2 as *contrastive estimation* (CE).

CE seeks to move probability mass from the neighborhood of an observed x_i to x_i itself. The learner hypothesizes that good models are those which discriminate an observed example from its neighborhood. Put another way, the learner assumes not only that x_i is good, but that x_i is locally optimal in example space ($\mathcal{X}$), and that alternative, similar examples (from the neighborhood) are inferior. Rather than explain all of the data, the model must only explain (using hidden variables) why the observed sentence is better than its neighbors. Of course, the validity of this hypothesis will depend on the form of the neighborhood function.

Consider, as a concrete example, learning natural language syntax. In Smith and Eisner (2005), we define a sentence's neighborhood to be a set of slightly-altered sentences that use the same lexemes, as suggested at the start of this section. While their syntax is degraded, the inferred meaning of any of these altered sentences is typically close to the intended meaning, yet the speaker *chose* x and not one of the other $x' \in \mathcal{N}(x)$. Why? Deletions are likely to violate subcategorization requirements, and transpositions are likely to violate word order requirements—both of which have something to do with syntax. x was the most grammatical option that conveyed the speaker's meaning, hence (we hope) roughly the most grammatical option in the neighborhood $\mathcal{N}(x)$, and the syntactic model should make it so.

3 Log-Linear Models

We have not yet specified the form of our probabilistic model, only that it is parameterized by $\vec{\theta} \in \mathbb{R}^n$. Log-linear models, which we will show are a natural fit for CE, assign probability to an (example, label) pair (x, y) according to

$$p\left(x, y \mid \vec{\theta}\right) \stackrel{\text{def}}{=} \frac{1}{Z\left(\vec{\theta}\right)} u\left(x, y \mid \vec{\theta}\right) \tag{3}$$

where the "unnormalized score" $u(x, y \mid \vec{\theta})$ is

$$u\left(x, y \mid \vec{\theta}\right) \stackrel{\text{def}}{=} \exp\left(\vec{\theta} \cdot \vec{f}(x, y)\right) \tag{4}$$

The notation above is defined as follows. $\vec{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n_{\geq 0}$ is a nonnegative vector feature function, and $\vec{\theta} \in \mathbb{R}^n$ are the corresponding feature weights (the model's parameters). Because the features can take any form and need not be orthogonal, log-linear models can capture arbitrary dependencies in the data and cleanly incorporate them into a model.

$Z(\vec{\theta})$ (the partition function) is chosen so that $\sum_{(x,y)} p(x, y \mid \vec{\theta}) = 1$; i.e., $Z(\vec{\theta}) = \sum_{(x,y)} u(x, y \mid \vec{\theta})$. u is typically easy to compute for a given (x, y), but Z may be much harder to compute. All the objective functions in this paper take the form

$$\prod_i \frac{\sum_{(x,y)\in\mathcal{A}_i} p\left(x, y \mid \vec{\theta}\right)}{\sum_{(x,y)\in\mathcal{B}_i} p\left(x, y \mid \vec{\theta}\right)} \tag{5}$$

likelihood criterion	$\mathcal{A}_i$	$\mathcal{B}_i$
joint	$\{(x_i, y_i^*)\}$	$\mathcal{X} \times \mathcal{Y}$
conditional	$\{(x_i, y_i^*)\}$	$\{x_i\} \times \mathcal{Y}$
marginal (*a là* EM)	$\{x_i\} \times \mathcal{Y}$	$\mathcal{X} \times \mathcal{Y}$
contrastive	$\{x_i\} \times \mathcal{Y}$	$\mathcal{N}(x_i) \times \mathcal{Y}$

Table 2: Supervised (upper box) and unsupervised (lower box) estimation with log-linear models in terms of Eq. 5.

where $\mathcal{A}_i \subset \mathcal{B}_i$ (for each i). For log-linear models this is simply

$$\prod_i \frac{\sum_{(x,y)\in\mathcal{A}_i} u\left(x, y \mid \vec{\theta}\right)}{\sum_{(x,y)\in\mathcal{B}_i} u\left(x, y \mid \vec{\theta}\right)} \tag{6}$$

So there is no need to compute $Z(\vec{\theta})$, but we *do* need to compute sums over $\mathcal{A}$ and $\mathcal{B}$. Tab. 2 summarizes some concrete examples; see also §3.1–3.2.

We would prefer to choose an objective function such that these sums are easy. CE focuses on choosing appropriate small contrast sets $\mathcal{B}_i$, both for efficiency and to guide the learner. The natural choice for $\mathcal{A}_i$ (which is usually easier to sum over) is the set of (x, y) that are consistent with what was observed (partially or completely) about the ith training example, i.e., the numerator $\sum_{(x,y)\in\mathcal{A}_i} p(x, y \mid \vec{\theta})$ is designed to find $p(\text{observation } i \mid \vec{\theta})$. The idea is to focus the probability mass within $\mathcal{B}_i$ on the subset $\mathcal{A}_i$ where the i the training example is known to be.

It is possible to build log-linear models where each x_i is a sequence.[2] In this paper, each model is a weighted finite-state automaton (WFSA) where states correspond to POS tags. The parameter vector $\vec{\theta} \in \mathbb{R}^n$ specifies a weight for each of the n transitions in the automaton. y is a hidden path through the automaton (determining a POS sequence), and x is the string it emits. $u(x, y \mid \vec{\theta})$ is defined by applying $\exp$ to the total weight of all transitions in y. This is an example of Eqs. 4 and 6 where $f_j(x, y)$ is the number of times the path y takes the jth transition.

The partition function $Z(\vec{\theta})$ of the WFSA is found by adding up the u-scores of all paths through the WFSA. For a k-state WFSA, this equates to solving a linear system of k equations in k variables (Tarjan, 1981). But if the WFSA contains cycles this infinite sum may diverge. Alternatives to exact com-

[2]These are exemplified by CRFs (Lafferty et al., 2001), which can be viewed alternately as undirected dynamic graphical models with a chain topology, as log-linear models over entire sequences with local features, or as WFSAs. Because "CRF" implies CL estimation, we use the term "WFSA."

putation, like random sampling (see, e.g., Abney, 1997), will not help to avoid this difficulty; in addition, convergence rates are in general unknown and bounds difficult to prove. We would prefer to sum over finitely many paths in $\mathcal{B}_i$.

3.1 Parameter estimation (supervised)

For log-linear models, both CL and JL estimation (Tab. 1) are available. In terms of Eq. 5, both set $\mathcal{A}_i = \{(x_i, y_i^*)\}$. The difference is in $\mathcal{B}$: for JL, $\mathcal{B}_i = \mathcal{X} \times \mathcal{Y}$, so summing over $\mathcal{B}_i$ is equivalent to computing the partition function $Z(\vec{\theta})$. Because that sum is typically difficult, CL is preferred; $\mathcal{B}_i = \{x_i\} \times \mathcal{Y}$ for x_i, which is often tractable. For sequence models like WFSAs it is computed using a dynamic programming algorithm (the forward algorithm for WFSAs). Klein and Manning (2002) argue for CL on grounds of accuracy, but see also Johnson (2001). See Tab. 2; other contrast sets $\mathcal{B}_i$ are also possible.

When $\mathcal{B}_i$ contains only x_i paired with the current best competitor ($\hat{y}$) to y_i^*, we have a technique that resembles maximum margin training (Crammer and Singer, 2001). Note that $\hat{y}$ will then change across training iterations, making $\mathcal{B}_i$ dynamic.

3.2 Parameter estimation (unsupervised)

The difference between supervised and unsupervised learning is that in the latter case, $\mathcal{A}_i$ is forced to sum over label sequences y because they weren't observed. In the unsupervised case, CE maximizes

$$\mathcal{L}_{\mathcal{N}}\left(\vec{\theta}\right) = \log \prod_i \frac{\sum_{y \in \mathcal{Y}} u\left(x_i, y \mid \vec{\theta}\right)}{\sum_{(x,y) \in \mathcal{N}(x_i) \times \mathcal{Y}} u\left(x, y \mid \vec{\theta}\right)} \tag{7}$$

In terms of Eq. 5, $\mathcal{A} = \{x_i\} \times \mathcal{Y}$ and $\mathcal{B} = \mathcal{N}(x_i) \times \mathcal{Y}$. EM's objective function (Eq. 1) is a special case where $\mathcal{N}(x_i) = \mathcal{X}$, for all i, and the denominator becomes $Z(\vec{\theta})$. An alternative is to restrict the neighborhood to the set of observed training examples rather than all possible examples (Riezler, 1999; Johnson et al., 1999; Riezler et al., 2000):

$$\prod_i \left[u\left(x_i \mid \vec{\theta}\right) \Big/ \sum_j u\left(x_j \mid \vec{\theta}\right) \right] \tag{8}$$

Viewed as a CE method, this approach (though effective when there are few hypotheses) seems misguided; the objective says to move mass to each example at the expense of all other training examples.

Another variant is *conditional* EM. Let x_i be a pair $(x_{i,1}, x_{i,2})$ and define the neighborhood to be $\mathcal{N}(x_i) = \{\bar{x} = (\bar{x}_1, x_{i,2})\}$. This approach has been applied to conditional densities (Jebara and Pentland, 1998) and conditional training of acoustic models with hidden variables (Valtchev et al., 1997).

Generally speaking, CE is equivalent to some kind of EM when $\mathcal{N}(\cdot)$ is an equivalence relation on examples, so that the neighborhoods partition $\mathcal{X}$. Then if q is any fixed (untrained) distribution over neighborhoods, CE equates to running EM on the model defined by

$$p'\left(x, y \mid \vec{\theta}\right) \stackrel{\text{def}}{=} q\left(\mathcal{N}(x)\right) \cdot p\left(x, y \mid \mathcal{N}(x), \vec{\theta}\right) \tag{9}$$

CE may also be viewed as an importance sampling approximation to EM, where the sample space $\mathcal{X}$ is replaced by $\mathcal{N}(x_i)$. We will demonstrate experimentally that CE is not just an approximation to EM; it makes sense from a modeling perspective.

In §4, we will describe neighborhoods of sequences that can be represented as acyclic *lattices* built directly from an observed sequence. The sum over $\mathcal{B}_i$ is then the total u-score in our model of all paths in the neighborhood lattice. To compute this, intersect the WFSA and the lattice, obtaining a new *acyclic* WFSA, and sum the u-scores of all its paths (Eisner, 2002) using a simple dynamic programming algorithm akin to the forward algorithm. The sum over $\mathcal{A}_i$ may be computed similarly.

CE with lattice neighborhoods is not confined to the WFSAs of this paper; when estimating weighted CFGs, the key algorithm is the inside algorithm for lattice parsing (Smith and Eisner, 2005).

3.3 Numerical optimization

To maximize the neighborhood likelihood (Eq. 7), we apply a standard numerical optimization method (L-BFGS) that iteratively climbs the function using knowledge of its value and gradient (Liu and Nocedal, 1989). The partial derivative of $\mathcal{L}_{\mathcal{N}}$ with respect to the jth feature weight θ_j is

$$\frac{\partial \mathcal{L}_{\mathcal{N}}}{\partial \theta_j} = \sum_i \mathbf{E}_{\vec{\theta}}\left[f_j \mid x_i\right] - \mathbf{E}_{\vec{\theta}}\left[f_j \mid \mathcal{N}(x_i)\right] \tag{10}$$

This looks similar to the gradient of log-linear likelihood functions on complete data, though the expectation on the left is in those cases replaced by an observed feature value $f_j(x_i, y_i^*)$. In this paper, the

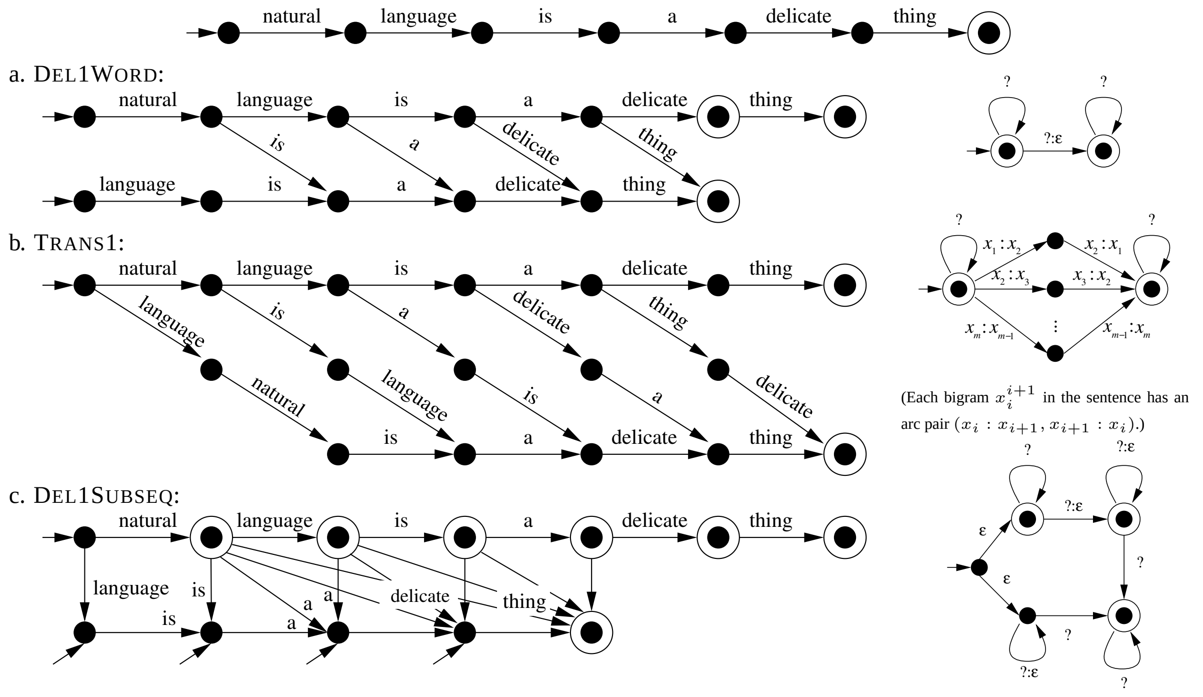

natural language is a delicate thing
a. DEL1WORD:
natural language is a delicate thing
is a delicate thing
language is a delicate thing
?
?:ε
b. TRANS1:
natural language is a delicate thing
language is a delicate thing
natural language is a delicate
is a delicate thing
?
$x_1 : x_2$
$x_2 : x_1$
$x_2 : x_3$
$x_3 : x_2$
$x_m : x_{m-1}$
$x_{m-1} : x_m$
(Each bigram x_i^{i+1} in the sentence has an arc pair $(x_i : x_{i+1}, x_{i+1} : x_i)$.)
c. DEL1SUBSEQ:
natural language is a delicate thing
language is a delicate thing
is a
ε
?:ε
?

Figure 1: A sentence and three lattices representing some of its neighborhoods. The transducer used to generate each neighborhood lattice (via composition with the sentence, followed by determinization and minimization) is shown to its right.

expectations in Eq. 10 are computed by the forward-backward algorithm generalized to lattices.

We emphasize that the function $\mathcal{L}_{\mathcal{N}}$ is not globally concave; our search will lead only to a local optimum.[3] Therefore, as with all unsupervised statistical learning, the bias in the initialization of $\vec{\theta}$ will affect the quality of the estimate and the performance of the method. In future we might wish to apply techniques for avoiding local optima, such as deterministic annealing (Smith and Eisner, 2004).

4 Lattice Neighborhoods

We next consider some non-classical neighborhood functions for sequences. When $\mathcal{X} = \Sigma^+$ for some symbol alphabet Σ, certain kinds of neighborhoods have natural, compact representations. Given an input string $x = \langle x_1, x_2, ..., x_m \rangle$, we write x_i^j for the substring $\langle x_i, x_{i+1}, ..., x_j \rangle$ and x_1^m for the whole string. Consider first the neighborhood consisting of all sequences generated by deleting a single symbol from the m-length sequence x_1^m:

$$\textsc{Del1Word}(x_1^m) \quad = \quad \left\{ x_1^{\ell-1} x_{\ell+1}^m \mid 1 \le \ell \le m \right\} \cup \{x_1^m\}$$

This set consists of $m + 1$ strings and can be compactly represented as a lattice (see Fig. 1a). Another neighborhood involves transposing any pair of adjacent words:

$$\textsc{Trans1}(x_1^m) = \left\{ x_1^{\ell-1} x_{\ell+1} x_\ell x_{\ell+2}^m \mid 1 \le \ell < m \right\} \cup \{x_1^m\}$$

This set can also be compactly represented as a lattice (Fig. 1b). We can combine DEL1WORD and TRANS1 by taking their union; this gives a larger neighborhood, DELORTRANS1.[4]

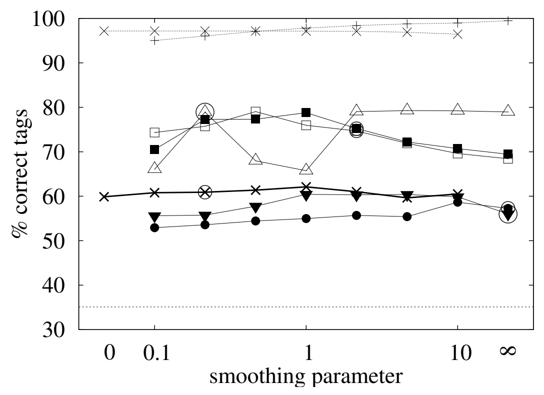
The DEL1SUBSEQ neighborhood allows the deletion of any contiguous subsequence of words that is strictly smaller than the whole sequence. This lattice is similar to that of DEL1WORD, but adds some arcs (Fig. 1c); the size of this neighborhood is $O(m^2)$.

A final neighborhood we will consider is LENGTH, which consists of Σ^m. CE with the LENGTH neighborhood is very similar to EM; it is equivalent to using EM to estimate the parameters of a model defined by Eq. 9 where q is any fixed (untrained) distribution over lengths.

When the vocabulary Σ is the set of words in a natural language, it is never fully known; approximations for defining LENGTH $= \Sigma^m$ include using observed Σ from the training set (as we do) or adding a special OOV symbol.

[3]Without any hidden variables, $\mathcal{L}_{\mathcal{N}}$ *is* globally concave.

[4]In general, the lattices are obtained by composing the observed sequence with a small FST and determinizing and minimizing the result; the relevant transducers are shown in Fig. 1.



	12K		24K		48K		96K	
	sel.	*oracle*	sel.	*oracle*	sel.	*oracle*	sel.	*oracle*
+ CRF (supervised)		*100.0*		*99.8*		*99.8*		*99.5*
× HMM (supervised)		*99.3*		*98.5*		*97.9*		*97.2*
△ LENGTH	**74.9**	***77.4***	**78.7**	***81.5***	**78.3**	***81.3***	**78.9**	***79.3***
■ DELORTR1	70.8	*70.8*	78.6	*78.6*	**78.3**	*79.1*	75.2	*78.8*
□ TRANS1	72.7	*72.7*	77.2	*77.2*	78.1	*79.4*	74.7	*79.0*
× EM	49.5	*52.9*	55.5	*58.0*	59.4	*60.9*	60.9	*62.1*
▼ DEL1WORD	55.4	*55.6*	58.6	*60.3*	59.9	*60.2*	59.9	*60.4*
● DEL1SSQ	53.0	*53.3*	55.0	*56.7*	55.3	*55.4*	57.3	*58.7*
random expected		*35.2*		*35.1*		*35.1*		*35.1*
ambiguous words		6,244		12,923		25,879		51,521

Figure 2: Percent ambiguous words tagged correctly in the 96K dataset, as the smoothing parameter (λ in the case of EM, σ^2 in the CE cases) varies. The model selected from each criterion using unlabeled development data is circled in the plot. Dataset size is varied in the table at right, which shows models selected using unlabeled development data ("sel.") and using an oracle ("*oracle*," the highest point on a curve). Across conditions, some neighborhood roughly splits the difference between supervised models and EM.

5 Experiments

We compare CE (using neighborhoods from §4) with EM on POS tagging using unlabeled data.

5.1 Comparison with EM

Our experiments are inspired by those in Merialdo (1994); we train a trigram tagger using only unlabeled data, assuming complete knowledge of the tagging dictionary.[5] In our experiments, we varied the amount of data available (12K–96K words of WSJ), the heaviness of smoothing, and the estimation criterion. In all cases, training stopped when the relative change in the criterion fell below 10^{-4} between steps (typically ≤ 100 steps). For this corpus and tag set, on average, a tagger must decide between 2.3 tags for a given token.

The generative model trained by EM was identical to Merialdo's: a second-order HMM. We smoothed using a flat Dirichlet prior with single parameter λ for all distributions (λ-values from 0 to 10 were tested).[6] The model was initialized uniformly.

The log-linear models trained by CE used the same feature set, though the feature weights are no longer log-probabilities and there are no sum-to-one constraints. In addition to an unsmoothed trial, we tried diagonal Gaussian priors (quadratic penalty) with σ^2 ranging from 0.1 to 10. The models were initialized with all $\theta_j = 0$.

Unsupervised model selection. For each (criterion, dataset) pair, we selected the smoothing trial that gave the highest estimation criterion score on a 5K-word development set (also unlabeled).

Results. The plot in Fig. 2 shows the Viterbi accuracy of each criterion trained on the 96K-word dataset as smoothing was varied; the table shows, for each (criterion, dataset) pair the performance of the selected λ or σ^2 and the one chosen by an oracle. LENGTH, TRANS1, and DELORTRANS1 are consistently the best, far out-stripping EM. These gains dwarf the performance of EM on over 1.1M words (66.6% as reported by Smith and Eisner (2004)), even when the latter uses improved search (70.0%). DEL1WORD and DEL1SUBSEQ, on the other hand, are poor, even worse than EM on larger datasets.

An important result is that neighborhoods do not succeed by virtue of *approximating* log-linear EM; if that were so, we would expect larger neighborhoods (like DEL1SUBSEQ) to out-perform smaller ones (like TRANS1)—this is not so. DEL1SUBSEQ and DEL1WORD are poor because they do not give helpful classes of negative evidence: deleting a word or a short subsequence often does very little damage. Put another way, models that do a good job of explaining why no word or subsequence should be deleted do not do so using the familiar POS categories.

The LENGTH neighborhood is as close to log-linear EM as it is practical to get. The inconsistencies in the LENGTH curve (Fig. 2) are notable and also appeared at the other training set sizes. Believing this might be indicative of brittleness in Viterbi label selection, we computed the *expected*

[5]Without a tagging dictionary, tag names are interchangeable and cannot be evaluated on gold-standard accuracy. We address the tagging dictionary assumption in §5.2.

[6]This is equivalent to add-λ smoothing within every M step.

words in tagging dict.	DelOrTrans1 trigram sel.	*oracle*	DelOrTrans1 trigram + spelling sel.	*oracle*	Trans1 trigram sel.	*oracle*	Trans1 trigram + spelling sel.	*oracle*	Length trigram sel.	*oracle*	Length trigram + spelling sel.	*oracle*	EM trigram sel.	*oracle*	*random expected*	*ambiguous words*	*ave. tags/token*
all train & dev.	78.3	*90.1*	80.9	*91.1*	**90.4**	*90.4*	88.7	*90.9*	87.8	*90.4*	87.1	***91.9***	78.0	*84.4*	*69.5*	13,150	2.3
1st 500 sents.	72.3	*84.8*	80.2	***90.8***	80.8	*82.9*	**88.1**	*90.1*	68.1	*78.3*	76.9	*83.2*	77.2	*80.5*	*60.5*	13,841	3.7
count ≥ 2	69.5	*81.3*	**79.5**	***90.3***	77.0	*78.6*	78.7	*90.1*	65.3	*75.2*	73.3	*73.8*	70.1	*70.9*	*56.6*	14,780	4.4
count ≥ 3	65.0	*77.2*	78.3	***89.8***	71.7	*73.4*	**78.4**	*89.5*	62.8	*72.3*	73.2	*73.6*	66.5	*66.5*	*51.0*	15,996	5.5

Table 3: Percent of *all* words correctly tagged in the 24K dataset, as the tagging dictionary is diluted. Unsupervised model selection ("sel.") and oracle model selection ("*oracle*") across smoothing parameters are shown. Note that we evaluated on *all* words (unlike Fig. 3) and used 17 coarse tags, giving higher scores than in Fig. 2.

accuracy of the LENGTH models; the same "dips" were present. This could indicate that the learner was trapped in a local maximum, suggesting that, since other criteria did not exhibit this behavior, LENGTH might be a bumpier objective surface. It would be interesting to measure the bumpiness (sensitivity to initial conditions) of different contrastive objectives.[7]

5.2 Removing knowledge, adding features

The assumption that the tagging dictionary is completely known is difficult to justify. While a POS lexicon might be available for a new language, certainly it will not give exhaustive information about all word types in a corpus. We experimented with removing knowledge from the tagging dictionary, thereby increasing the difficulty of the task, to see how well various objective functions could recover. One means to recovery is the addition of features to the model—this is easy with log-linear models but not with classical generative models.

We compared the performance of the best neighborhoods (LENGTH, DELORTRANS1, and TRANS1) from the first experiment, plus EM, using three *diluted* dictionaries and the original one, on the 24K dataset. A diluted dictionary adds (tag, word) entries so that rare words are allowed with *any* tag, simulating zero prior knowledge about the word. "Rare" might be defined in different ways; we used three definitions: words unseen in the first 500 sentences (about half of the 24K training corpus); singletons (words with count ≤ 1); and words with count ≤ 2. To allow more trials, we projected the original 45 tags onto a coarser set of 17 (e.g., RB$*$ $\rightarrow$ADV).

To take better advantage of the power of log-linear models—specifically, their ability to incorporate novel features—we also ran trials augmenting the model with *spelling* features, allowing exploitation of correlations between *parts* of the word and a possible tag. Our spelling features included all observed 1-, 2-, and 3-character suffixes, initial capitalization, containing a hyphen, and containing a digit.

Results. Fig. 3 plots tagging accuracy (on ambiguous words) for each dictionary on the 24K dataset. The x-axis is the smoothing parameter (λ for EM, σ^2 for CE). Note that the different plots are not comparable, because their y-axes are based on different sets of ambiguous words.

So that models under different dilution conditions could be compared, we computed accuracy on *all* words; these are shown in Tab. 3. The reader will notice that there is often a large gap between unsupervised and oracle model selection; this draws attention to a need for better unsupervised regularization and model selection techniques.

Without spelling features, all models perform worse as knowledge is removed. But LENGTH suffers most substantially, relative to its initial performance. Why is this? LENGTH (like EM) requires the model to explain why a given sentence was seen instead of some other sentence of the same length. One way to make this explanation is to manipulate emission weights (i.e., for (tag, word) features): the learner can construct a good class-based *unigram* model of the text (where classes are tags). This is good for the LENGTH objective, but not for learning good POS tag sequences.

In contrast, DELORTRANS1 and TRANS1 do not allow the learner to manipulate emission weights for words not in the sentence. The sentence's goodness must be explained in a way other than by the words it contains: namely through the POS tags. To

[7] A reviewer suggested including a table comparing different criterion values for each learned model (i.e., each neighborhood evaluated on each other neighborhood). This table contained no big surprises; we note only that most models were the best on their own criterion, and among unsupervised models, LENGTH performed best on the CL criterion.

check this intuition, we built local normalized models $p(\text{word} \mid \text{tag})$ from the parameters learned by TRANS1 and LENGTH. For each tag, these were compared by KL divergence to the empirical lexical distributions (from labeled data). For the ten tags accounting for 95.6% of the data, LENGTH more closely matched the empirical lexical distributions. LENGTH is learning a correct distribution, but that distribution is not helpful for the task.

The improvement from adding spelling features is striking: DELORTRANS1 and TRANS1 recover nearly completely (modulo the model selection problem) from the diluted dictionaries. LENGTH sees far less recovery. Hence even our improved feature sets cannot compensate for the choice of neighborhood. This highlights our argument that a neighborhood is not an approximation to log-linear EM; LENGTH tries very hard to approximate log-linear EM but requires a good dictionary to be on par with the other criteria. Good neighborhoods, rather, perform well in their own right.

6 Future Work

Foremost for future work is the "minimally supervised" paradigm in which a small amount of labeled data is available (see, e.g., Clark et al. (2003)). Unlike well-known "bootstrapping" approaches (Yarowsky, 1995), EM and CE have the possible advantage of maintaining posteriors over hidden labels (or structure) throughout learning; bootstrapping either chooses, for each example, a single label, or remains completely agnostic. One can envision a *mixed* objective function that tries to fit the labeled examples while discriminating unlabeled examples from their neighborhoods.[8]

Regardless of how much (if any) data are labeled, the question of good smoothing techniques requires more attention. Here we used a single zero-mean, constant-variance Gaussian prior for all parameters. Better performance might be achieved by allowing different variances for different feature types. This

[8]Zhu and Ghahramani (2002) explored the semi-supervised classification problem for spatially-distributed data, where some data are labeled, using a Boltzmann machine to model the dataset. For them, the Markov random field is over labeling configurations for all examples, not, as in our case, complex structured labels for a particular example. Hence their $\mathcal{B}$ (Eq. 5), though very large, was finite and could be sampled.

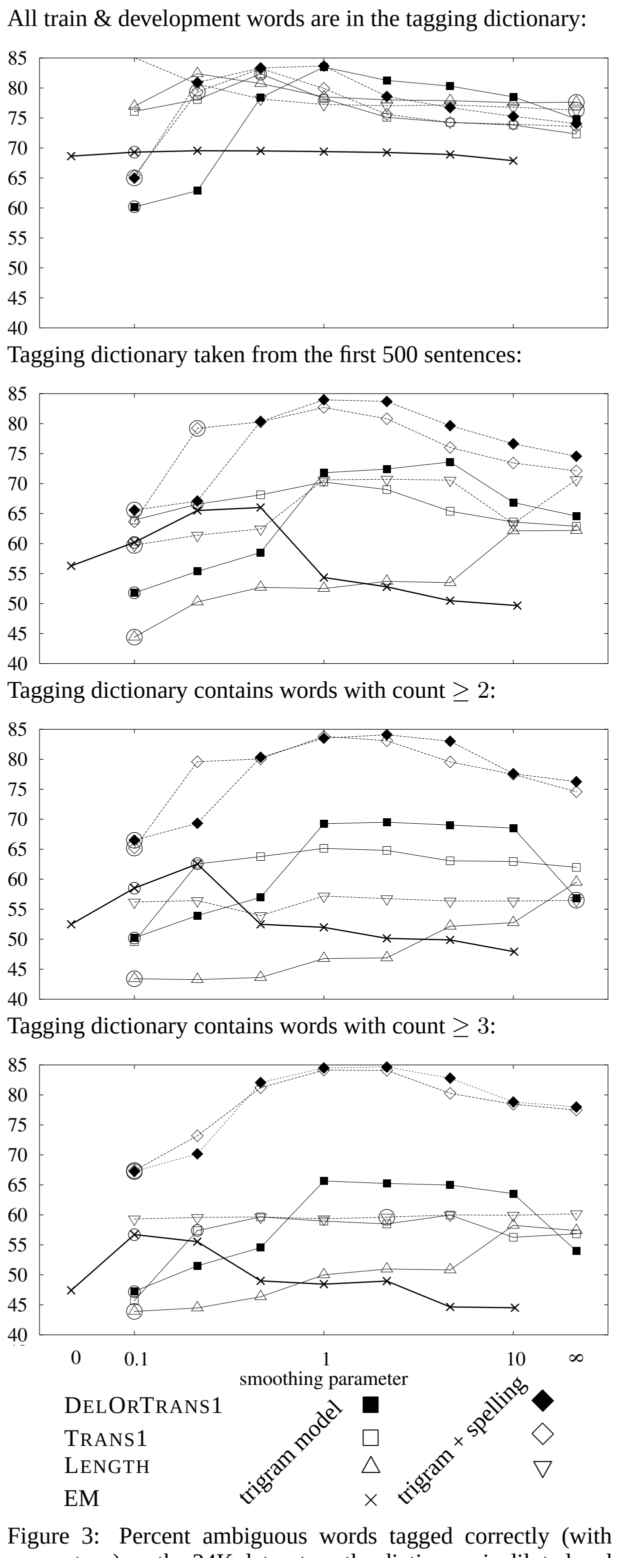


Figure 3: Percent ambiguous words tagged correctly (with coarse tags) on the 24K dataset, as the dictionary is diluted and with spelling features. Each graph corresponds to a different level of dilution. Models selected using unlabeled development data are circled. These plots (unlike Tab. 3) are *not* comparable to each other because each is measured on a different set of ambiguous words.

leads to a need for more efficient tuning of the prior parameters on development data.

The effectiveness of CE (and different neighborhoods) for dependency grammar induction is explored in Smith and Eisner (2005) with considerable success. We introduce there the notion of designing neighborhoods to guide learning for particular tasks. Instead of guiding an unsupervised learner to match linguists' annotations, the choice of neighborhood might be made to direct the learner toward hidden structure that is helpful for error-correction tasks like spelling correction and punctuation restoration that may benefit from a grammatical model.

Wang et al. (2002) discuss the latent maximum entropy principle. They advocate running EM many times and selecting the local maximum that maximizes entropy. One might do the same for the local maxima of any CE objective, though theoretical and experimental support for this idea remain for future work.

7 Conclusion

We have presented *contrastive estimation*, a new probabilistic estimation criterion that forces a model to explain why the given training data were better than bad data implied by the positive examples. We have shown that for unsupervised sequence modeling, this technique is efficient and drastically outperforms EM; for POS tagging, the gain in accuracy over EM is twice what we would get from ten times as much data and improved search, sticking with EM's criterion (Smith and Eisner, 2004). On this task, with certain neighborhoods, contrastive estimation suffers less than EM does from diminished prior knowledge and is able to exploit new features—that EM can't—to largely recover from the loss of knowledge.

References

S. P. Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–617.

Y. Altun, M. Johnson, and T. Hofmann. 2003. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proc. of EMNLP*.

E. Charniak. 1993. *Statistical Language Learning*. MIT Press.

S. Clark, J. R. Curran, and M. Osborne. 2003. Bootstrapping POS taggers using unlabelled data. In *Proc. of CoNLL*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

K. Crammer and Y. Singer. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(5):265–92.

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. Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*.

G. E. Hinton. 2003. Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, University College London.

T. Jebara and A. Pentland. 1998. Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Proc. of NIPS*.

M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proc. of ACL*.

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Proc. of ACL*.

B.-H. Juang and S. Katagiri. 1992. Discriminative learning for minimum error classification. *IEEE Trans. Signal Processing*, 40:3043–54.

D. Klein and C. D. Manning. 2002. Conditional structure vs. conditional estimation in NLP models. 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*.

D. C. Liu and J. Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–28.

A. McCallum and W. Li. 2003. Early results for named-entity extraction with conditional random fields. In *Proc. of CoNLL*.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–72.

Y. Miyao and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proc. of HLT*.

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*.

S. Riezler. 1999. *Probabilistic Constraint Logic Programming*. Ph.D. thesis, Universität Tübingen.

R. Rosenfeld. 1994. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. thesis, CMU.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*.

N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*.

R. E. Tarjan. 1981. A unified approach to path problems. *Journal of the ACM*, 28(3):577–93.

V. Valtchev, J. J. Odell, P. C. Woodland, and S. J. Young. 1997. MMIE training of large vocabulary speech recognition systems. *Speech Communication*, 22(4):303–14.

S. Wang, R. Rosenfeld, Y. Zhao, and D. Schuurmans. 2002. The latent maximum entropy principle. In *Proc. of ISIT*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.

X. Zhu and Z. Ghahramani. 2002. Towards semi-supervised classification with Markov random fields. Technical Report CMU-CALD-02-106, Carnegie Mellon University.

Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling

Jenny Rose Finkel, Trond Grenager, and Christopher Manning
Computer Science Department
Stanford University
Stanford, CA 94305
{jrfinkel, grenager, mannning}@cs.stanford.edu

Abstract

Most current statistical natural language processing models use only local features so as to permit dynamic programming in inference, but this makes them unable to fully account for the long distance structure that is prevalent in language use. We show how to solve this dilemma with *Gibbs sampling*, a simple Monte Carlo method used to perform approximate inference in factored probabilistic models. By using simulated annealing in place of Viterbi decoding in sequence models such as HMMs, CMMs, and CRFs, it is possible to incorporate non-local structure while preserving tractable inference. We use this technique to augment an existing CRF-based information extraction system with long-distance dependency models, enforcing label consistency and extraction template consistency constraints. This technique results in an error reduction of up to 9% over state-of-the-art systems on two established information extraction tasks.

1 Introduction

Most statistical models currently used in natural language processing represent only local structure. Although this constraint is critical in enabling tractable model inference, it is a key limitation in many tasks, since natural language contains a great deal of non-local structure. A general method for solving this problem is to relax the requirement of exact inference, substituting approximate inference algorithms instead, thereby permitting tractable inference in models with non-local structure. One such algorithm is *Gibbs sampling*, a simple Monte Carlo algorithm that is appropriate for inference in any factored probabilistic model, including sequence models and probabilistic context free grammars (Geman and Geman, 1984). Although Gibbs sampling is widely used elsewhere, there has been extremely little use of it in natural language processing.[1] Here, we use it to add non-local dependencies to sequence models for information extraction.

Statistical hidden state sequence models, such as Hidden Markov Models (HMMs) (Leek, 1997; Freitag and McCallum, 1999), Conditional Markov Models (CMMs) (Borthwick, 1999), and Conditional Random Fields (CRFs) (Lafferty et al., 2001) are a prominent recent approach to information extraction tasks. These models all encode the Markov property: decisions about the state at a particular position in the sequence can depend only on a small local window. It is this property which allows tractable computation: the Viterbi, Forward Backward, and Clique Calibration algorithms all become intractable without it.

However, information extraction tasks can benefit from modeling non-local structure. As an example, several authors (see Section 8) mention the value of enforcing label consistency in named entity recognition (NER) tasks. In the example given in Figure 1, the second occurrence of the token *Tanjug* is mislabeled by our CRF-based statistical NER system, because by looking only at local evidence it is unclear whether it is a person or organization. The first occurrence of *Tanjug* provides ample evidence that it is an organization, however, and by enforcing label consistency the system should be able to get it right. We show how to incorporate constraints of this form into a CRF model by using Gibbs sampling instead of the Viterbi algorithm as our inference procedure, and demonstrate that this technique yields significant improvements on two established IE tasks.

[1]Prior uses in NLP of which we are aware include: Kim et al. (1995), Della Pietra et al. (1997) and Abney (1997).

Proceedings of the 43rd Annual Meeting of the ACL, pages 363–370,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

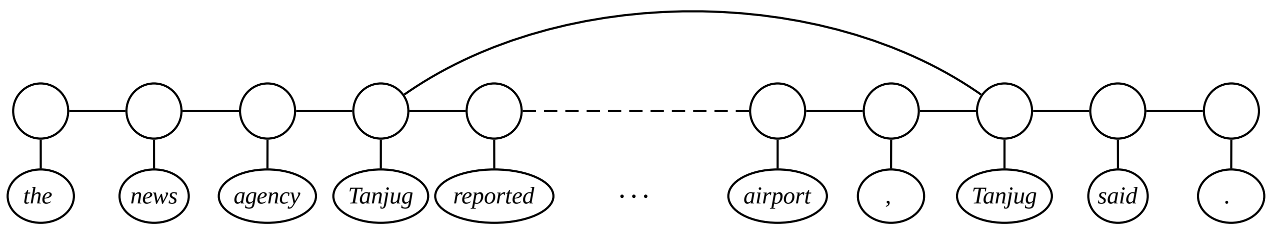


Figure 1: An example of the label consistency problem excerpted from a document in the CoNLL 2003 English dataset.

2 Gibbs Sampling for Inference in Sequence Models

In hidden state sequence models such as HMMs, CMMs, and CRFs, it is standard to use the Viterbi algorithm, a dynamic programming algorithm, to infer the most likely hidden state sequence given the input and the model (see, e.g., Rabiner (1989)). Although this is the only tractable method for exact computation, there are other methods for computing an approximate solution. Monte Carlo methods are a simple and effective class of methods for approximate inference based on sampling. Imagine we have a hidden state sequence model which defines a probability distribution over state sequences conditioned on any given input. With such a model M we should be able to compute the conditional probability $\mathrm{P}_M(\mathbf{s}|\mathbf{o})$ of any state sequence $\mathbf{s} = \{s_0, \ldots, s_N\}$ given some observed input sequence $\mathbf{o} = \{o_0, \ldots, o_N\}$. One can then sample sequences from the conditional distribution defined by the model. These samples are likely to be in high probability areas, increasing our chances of finding the maximum. The challenge is how to sample sequences efficiently from the conditional distribution defined by the model.

Gibbs sampling provides a clever solution (Geman and Geman, 1984). Gibbs sampling defines a Markov chain in the space of possible variable assignments (in this case, hidden state sequences) such that the stationary distribution of the Markov chain *is* the joint distribution over the variables. Thus it is called a Markov Chain Monte Carlo (MCMC) method; see Andrieu et al. (2003) for a good MCMC tutorial. In practical terms, this means that we can walk the Markov chain, occasionally outputting samples, and that these samples are guaranteed to be drawn from the target distribution. Furthermore, the chain is defined in very simple terms: from each state sequence we can only transition to a state sequence obtained by changing the state at any one position i, and the distribution over these possible transitions is just

$$\mathrm{P}_G(\mathbf{s}^{(t)}|\mathbf{s}^{(t-1)}) = \mathrm{P}_M(s_i^{(t)}|\mathbf{s}_{-i}^{(t-1)}, \mathbf{o}). \qquad (1)$$

where $\mathbf{s}_{-i}$ is all states except s_i. In other words, the transition probability of the Markov chain is the conditional distribution of the label at the position given the rest of the sequence. This quantity is easy to compute in any Markov sequence model, including HMMs, CMMs, and CRFs. One easy way to walk the Markov chain is to loop through the positions i from 1 to N, and for each one, to resample the hidden state at that position from the distribution given in Equation 1. By outputting complete sequences at regular intervals (such as after resampling all N positions), we can sample sequences from the conditional distribution defined by the model.

This is still a gravely inefficient process, however. Random sampling may be a good way to estimate the shape of a probability distribution, but it is not an efficient way to do what we want: find the maximum. However, we cannot just transition greedily to higher probability sequences at each step, because the space is extremely non-convex. We can, however, borrow a technique from the study of non-convex optimization and use *simulated annealing* (Kirkpatrick et al., 1983). Geman and Geman (1984) show that it is easy to modify a Gibbs Markov chain to do annealing; at time t we replace the distribution in (1) with

$$\mathrm{P}_A(\mathbf{s}^{(t)}|\mathbf{s}^{(t-1)}) = \frac{\mathrm{P}_M(s_i^{(t)}|\mathbf{s}_{-i}^{(t-1)}, \mathbf{o})^{1/c_t}}{\sum_j \mathrm{P}_M(s_j^{(t)}|\mathbf{s}_{-j}^{(t-1)}, \mathbf{o})^{1/c_t}} \qquad (2)$$

where $\mathbf{c} = \{c_0, \ldots, c_T\}$ defines a *cooling schedule*. At each step, we raise each value in the conditional distribution to an exponent and renormalize before sampling from it. Note that when $c = 1$ the distribution is unchanged, and as $c \to 0$ the distribution

Inference	CoNLL	Seminars
Viterbi	85.51	91.85
Gibbs Sampling	85.54	91.85
	85.51	91.85
	85.49	91.85
	85.51	91.85
	85.51	91.85
	85.51	91.85
	85.51	91.85
	85.51	91.85
	85.51	91.86
Mean	85.51	91.85
Std. Dev.	0.01	0.004

Table 1: An illustration of the effectiveness of Gibbs sampling, compared to Viterbi inference, for the two tasks addressed in this paper: the CoNLL named entity recognition task, and the CMU Seminar Announcements information extraction task. We show 10 runs of Gibbs sampling in the same CRF model that was used for Viterbi. For each run the sampler was initialized to a random sequence, and used a linear annealing schedule that sampled the complete sequence 1000 times. CoNLL performance is measured as per-entity F_1, and CMU Seminar Announcements performance is measured as per-token F_1.

becomes sharper, and when $c = 0$ the distribution places all of its mass on the maximal outcome, having the effect that the Markov chain always climbs uphill. Thus if we gradually decrease c from 1 to 0, the Markov chain increasingly tends to go uphill. This annealing technique has been shown to be an effective technique for stochastic optimization (Laarhoven and Arts, 1987).

To verify the effectiveness of Gibbs sampling and simulated annealing as an inference technique for hidden state sequence models, we compare Gibbs and Viterbi inference methods for a basic CRF, without the addition of any non-local model. The results, given in Table 1, show that if the Gibbs sampler is run long enough, its accuracy is the same as a Viterbi decoder.

3 A Conditional Random Field Model

Our basic CRF model follows that of Lafferty et al. (2001). We choose a CRF because it represents the state of the art in sequence modeling, allowing both discriminative training and the bi-directional flow of probabilistic information across the sequence. A CRF is a conditional sequence model which represents the probability of a hidden state sequence given some observations. In order to facilitate obtaining the conditional probabilities we need for Gibbs sampling, we generalize the CRF model in a way that is consistent with the Markov Network literature (see Cowell et al. (1999)): we create a linear chain of *cliques*, where each clique, c, represents the probabilistic relationship between an adjacent pair of states[2] using a *clique potential* ϕ_c, which is just a table containing a value for each possible state assignment. The table is not a true probability distribution, as it only accounts for local interactions within the clique. The clique potentials themselves are defined in terms of exponential models conditioned on features of the observation sequence, and must be instantiated for each new observation sequence. The sequence of potentials in the clique chain then defines the probability of a state sequence (given the observation sequence) as

$$\mathrm{P}_{\mathrm{CRF}}(\mathbf{s}|\mathbf{o}) \propto \prod_{i=1}^{N} \phi_i(s_{i-1}, s_i) \qquad (3)$$

where $\phi_i(s_{i-1}, s_i)$ is the element of the clique potential at position i corresponding to states s_{i-1} and s_i.[3]

Feature	NER	TF
Current Word	✓	✓
Previous Word	✓	✓
Next Word	✓	✓
Current Word Character n-gram	all	length ≤ 6
Current POS Tag	✓	
Surrounding POS Tag Sequence	✓	
Current Word Shape	✓	✓
Surrounding Word Shape Sequence	✓	✓
Presence of Word in Left Window	size 4	size 9
Presence of Word in Right Window	size 4	size 9

Table 2: Features used by the CRF for the two tasks: named entity recognition (NER) and template filling (TF).

Although a full treatment of CRF training is beyond the scope of this paper (our technique assumes the model is already trained), we list the features used by our CRF for the two tasks we address in Table 2. During training, we regularized our exponential models with a quadratic prior and used the quasi-Newton method for parameter optimization. As is customary, we used the Viterbi algorithm to infer the most likely state sequence in a CRF.

[2]CRFs with larger cliques are also possible, in which case the potentials represent the relationship between a subsequence of k adjacent states, and contain $|S|^k$ elements.

[3]To handle the start condition properly, imagine also that we define a distinguished start state s_0.

The clique potentials of the CRF, instantiated for some observation sequence, can be used to easily compute the conditional distribution over states at a position given in Equation 1. Recall that at position i we want to condition on the states in the rest of the sequence. The state at this position can be influenced by any other state that it shares a clique with; in particular, when the clique size is 2, there are 2 such cliques. In this case the Markov blanket of the state (the minimal set of states that renders a state conditionally independent of all other states) consists of the two neighboring states and the observation sequence, all of which are observed. The conditional distribution at position i can then be computed simply as

$$\mathrm{P}_{\mathrm{CRF}}(s_i|\mathbf{s}_{-i},\mathbf{o}) \propto \phi_i(s_{i-1},s_i)\phi_{i+1}(s_i,s_{i+1}) \quad (4)$$

where the factor tables F in the clique chain are already conditioned on the observation sequence.

4 Datasets and Evaluation

We test the effectiveness of our technique on two established datasets: the CoNLL 2003 English named entity recognition dataset, and the CMU Seminar Announcements information extraction dataset.

4.1 The CoNLL NER Task

This dataset was created for the shared task of the Seventh Conference on Computational Natural Language Learning (CoNLL),[4] which concerned named entity recognition. The English data is a collection of Reuters newswire articles annotated with four entity types: *person* (PER), *location* (LOC), *organization* (ORG), and *miscellaneous* (MISC). The data is separated into a training set, a development set (testa), and a test set (testb). The training set contains 945 documents, and approximately 203,000 tokens. The development set has 216 documents and approximately 51,000 tokens, and the test set has 231 documents and approximately 46,000 tokens.

We evaluate performance on this task in the manner dictated by the competition so that results can be properly compared. Precision and recall are evaluated on a per-entity basis (and combined into an F_1 score). There is no partial credit; an incorrect entity boundary is penalized as both a false positive and as a false negative.

4.2 The CMU Seminar Announcements Task

This dataset was developed as part of Dayne Freitag's dissertation research Freitag (1998).[5] It consists of 485 emails containing seminar announcements at Carnegie Mellon University. It is annotated for four fields: *speaker*, *location*, *start time*, and *end time*. Sutton and McCallum (2004) used 5-fold cross validation when evaluating on this dataset, so we obtained and used their data splits, so that results can be properly compared. Because the entire dataset is used for testing, there is no development set. We also used their evaluation metric, which is slightly different from the method for CoNLL data. Instead of evaluating precision and recall on a per-entity basis, they are evaluated on a per-token basis. Then, to calculate the overall F_1 score, the F_1 scores for each class are averaged.

5 Models of Non-local Structure

Our models of non-local structure are themselves just sequence models, defining a probability distribution over all possible state sequences. It is possible to flexibly model various forms of constraints in a way that is sensitive to the linguistic structure of the data (e.g., one can go beyond imposing just exact identity conditions). One could imagine many ways of defining such models; for simplicity we use the form

$$\mathrm{P}_M(\mathbf{s}|\mathbf{o}) \propto \prod_{\lambda\in\Lambda} \theta_\lambda^{\#(\lambda,\mathbf{s},\mathbf{o})} \quad (5)$$

where the product is over a set of violation types Λ, and for each violation type λ we specify a penalty parameter θ_λ. The exponent $\#(\lambda,\mathbf{s},\mathbf{o})$ is the count of the number of times that the violation λ occurs in the state sequence $\mathbf{s}$ with respect to the observation sequence $\mathbf{o}$. This has the effect of assigning sequences with more violations a lower probability. The particular violation types are defined specifically for each task, and are described in the following two sections.

This model, as defined above, is not normalized, and clearly it would be expensive to do so. This

[4]Available at *http://cnts.uia.ac.be/conll2003/ner/*.

[5]Available at *http://nlp.shef.ac.uk/dot.kom/resources.html*.

	PER	LOC	ORG	MISC
PER	3141	4	5	0
LOC		6436	188	3
ORG			2975	0
MISC				2030

Table 3: Counts of the number of times multiple occurrences of a token sequence is labeled as different entity types in the same document. Taken from the CoNLL training set.

	PER	LOC	ORG	MISC
PER	1941	5	2	3
LOC	0	167	6	63
ORG	22	328	819	191
MISC	14	224	7	365

Table 4: Counts of the number of times an entity sequence is labeled differently from an occurrence of a subsequence of it elsewhere in the document. Rows correspond to sequences, and columns to subsequences. Taken from the CoNLL training set.

doesn't matter, however, because we only use the model for Gibbs sampling, and so only need to compute the conditional distribution at a single position i (as defined in Equation 1). One (inefficient) way to compute this quantity is to enumerate all possible sequences differing only at position i, compute the score assigned to each by the model, and renormalize. Although it seems expensive, this computation can be made very efficient with a straightforward memoization technique: at all times we maintain data structures representing the relationship between entity labels and token sequences, from which we can quickly compute counts of different types of violations.

5.1 CoNLL Consistency Model

Label consistency structure derives from the fact that within a particular document, different occurrences of a particular token sequence are unlikely to be labeled as different entity types. Although any one occurrence may be ambiguous, it is unlikely that all instances are unclear when taken together.

The CoNLL training data empirically supports the strength of the label consistency constraint. Table 3 shows the counts of entity labels for each pair of identical token sequences within a document, where both are labeled as an entity. Note that inconsistent labelings are very rare.[6] In addition, we also want to model subsequence constraints: having seen *Geoff Woods* earlier in a document as a person is a good indicator that a subsequent occurrence of *Woods* should also be labeled as a person. However, if we examine all cases of the labelings of other occurrences of subsequences of a labeled entity, we find that the consistency constraint does not hold nearly so strictly in this case. As an example, one document contains references to both *The China Daily*, a newspaper, and *China*, the country. Counts of subsequence labelings within a document are listed in Table 4. Note that there are many off-diagonal entries: the *China Daily* case is the most common, occurring 328 times in the dataset.

The penalties used in the long distance constraint model for CoNLL are the Empirical Bayes estimates taken directly from the data (Tables 3 and 4), except that we change counts of 0 to be 1, so that the distribution remains positive. So the estimate of a PER also being an ORG is $\frac{5}{3151}$; there were 5 instance of an entity being labeled as both, PER appeared 3150 times in the data, and we add 1 to this for smoothing, because PER-MISC never occured. However, when we have a phrase labeled differently in two different places, continuing with the PER-ORG example, it is unclear if we should penalize it as PER that is also an ORG or an ORG that is also a PER. To deal with this, we multiply the square roots of each estimate together to form the penalty term. The penalty term is then multiplied in a number of times equal to the length of the offending entity; this is meant to "encourage" the entity to shrink.[7] For example, say we have a document with three entities, *Rotor Volgograd* twice, once labeled as PER and once as ORG, and *Rotor*, labeled as an ORG. The likelihood of a PER also being an ORG is $\frac{5}{3151}$, and of an ORG also being a PER is $\frac{5}{3169}$, so the penalty for this violation is $(\sqrt{\frac{5}{3151}} \times \sqrt{\frac{5}{3151}})^2$. The likelihood of a ORG being a subphrase of a PER is $\frac{2}{842}$. So the total penalty would be $\frac{5}{3151} \times \frac{5}{3169} \times \frac{2}{842}$.

[6]A notable exception is the labeling of the same text as both organization and location within the same document. This is a consequence of the large portion of sports news in the CoNLL dataset, so that city names are often also team names.

[7]While there is no theoretical justification for this, we found it to work well in practice.

5.2 CMU Seminar Announcements Consistency Model

Due to the lack of a development set, our consistency model for the CMU Seminar Announcements is much simpler than the CoNLL model, the numbers where selected due to our intuitions, and we did not spend much time hand optimizing the model. Specifically, we had three constraints. The first is that all entities labeled as *start time* are normalized, and are penalized if they are inconsistent. The second is a corresponding constraint for end times. The last constraint attempts to consistently label the speakers. If a phrase is labeled as a *speaker*, we assume that the last word is the speaker's last name, and we penalize for each occurrance of that word which is not also labeled *speaker*. For the start and end times the penalty is multiplied in based on how many words are in the entity. For the speaker, the penalty is only multiplied in once. We used a hand selected penalty of $\exp -4.0$.

6 Combining Sequence Models

In the previous section we defined two models of non-local structure. Now we would like to incorporate them into the local model (in our case, the trained CRF), and use Gibbs sampling to find the most likely state sequence. Because both the trained CRF and the non-local models are themselves sequence models, we simply combine the two models into a *factored* sequence model of the following form

$$\mathrm{P}_F(\mathbf{s}|\mathbf{o}) \propto \mathrm{P}_M(\mathbf{s}|\mathbf{o})\mathrm{P}_L(\mathbf{s}|\mathbf{o}) \qquad (6)$$

where M is the local CRF model, L is the new non-local model, and F is the factored model.[8] In this form, the probability again looks difficult to compute (because of the normalizing factor, a sum over all hidden state sequences of length N). However, since we are only using the model for Gibbs sampling, we never need to compute the distribution explicitly. Instead, we need only the conditional probability of each position in the sequence, which can be computed as

$$\mathrm{P}_F(s_i|\mathbf{s}_{-i},\mathbf{o}) \propto \mathrm{P}_M(s_i|\mathbf{s}_{-i},\mathbf{o})\mathrm{P}_L(s_i|\mathbf{s}_{-i},\mathbf{o}). \quad (7)$$

[8]This model double-generates the state sequence conditioned on the observations. In practice we don't find this to be a problem.

CoNLL					
Approach	LOC	ORG	MISC	PER	ALL
B&M LT-RMN	–	–	–	–	80.09
B&M GLT-RMN	–	–	–	–	82.30
Local+Viterbi	88.16	80.83	78.51	90.36	85.51
NonLoc+Gibbs	88.51	81.72	80.43	92.29	86.86

Table 5: F_1 scores of the local CRF and non-local models on the CoNLL 2003 named entity recognition dataset. We also provide the results from Bunescu and Mooney (2004) for comparison.

CMU Seminar Announcements					
Approach	STIME	ETIME	SPEAK	LOC	ALL
S&M CRF	97.5	97.5	88.3	77.3	90.2
S&M Skip-CRF	96.7	97.2	88.1	80.4	90.6
Local+Viterbi	96.67	97.36	83.39	89.98	91.85
NonLoc+Gibbs	97.11	97.89	84.16	90.00	92.29

Table 6: F_1 scores of the local CRF and non-local models on the CMU Seminar Announcements dataset. We also provide the results from Sutton and McCallum (2004) for comparison.

At inference time, we then sample from the Markov chain defined by this transition probability.

7 Results and Discussion

In our experiments we compare the impact of adding the non-local models with Gibbs sampling to our baseline CRF implementation. In the CoNLL named entity recognition task, the non-local models increase the F_1 accuracy by about 1.3%. Although such gains may appear modest, note that they are achieved relative to a near state-of-the-art NER system: the winner of the CoNLL English task reported an F_1 score of 88.76. In contrast, the increases published by Bunescu and Mooney (2004) are relative to a baseline system which scores only 80.9% on the same task. Our performance is similar on the CMU Seminar Announcements dataset. We show the per-field F_1 results that were reported by Sutton and McCallum (2004) for comparison, and note that we are again achieving gains against a more competitive baseline system.

For all experiments involving Gibbs sampling, we used a linear cooling schedule. For the CoNLL dataset we collected 200 samples per trial, and for the CMU Seminar Announcements we collected 100 samples. We report the average of all trials, and in all cases we outperform the baseline with greater than 95% confidence, using the standard t-test. The trials had low standard deviations - 0.083% and 0.007% - and high minimun F-scores - 86.72%, and 92.28%

- for the CoNLL and CMU Seminar Announcements respectively, demonstrating the stability of our method.

The biggest drawback to our model is the computational cost. Taking 100 samples dramatically increases test time. Averaged over 3 runs on both Viterbi and Gibbs, CoNLL testing time increased from 55 to 1738 seconds, and CMU Seminar Announcements testing time increases from 189 to 6436 seconds.

8 Related Work

Several authors have successfully incorporated a label consistency constraint into probabilistic sequence model named entity recognition systems. Mikheev et al. (1999) and Finkel et al. (2004) incorporate label consistency information by using adhoc multi-stage labeling procedures that are effective but special-purpose. Malouf (2002) and Curran and Clark (2003) condition the label of a token at a particular position on the label of the most recent previous instance of that same token in a prior sentence of the same document. Note that this violates the Markov property, but is achieved by slightly relaxing the requirement of exact inference. Instead of finding the maximum likelihood sequence over the entire document, they classify one sentence at a time, allowing them to condition on the maximum likelihood sequence of previous sentences. This approach is quite effective for enforcing label consistency in many NLP tasks, however, it permits a forward flow of information only, which is not sufficient for all cases of interest. Chieu and Ng (2002) propose a solution to this problem: for each token, they define additional features taken from other occurrences of the same token in the document. This approach has the added advantage of allowing the training procedure to automatically learn good weightings for these "global" features relative to the local ones. However, this approach cannot easily be extended to incorporate other types of non-local structure.

The most relevant prior works are Bunescu and Mooney (2004), who use a *Relational Markov Network* (RMN) (Taskar et al., 2002) to explicitly models long-distance dependencies, and Sutton and McCallum (2004), who introduce *skip-chain CRFs*, which maintain the underlying CRF sequence model (which (Bunescu and Mooney, 2004) lack) while adding *skip edges* between distant nodes. Unfortunately, in the RMN model, the dependencies must be defined in the model structure before doing any inference, and so the authors use crude heuristic part-of-speech patterns, and then add dependencies between these text spans using *clique templates*. This generates a extremely large number of overlapping candidate entities, which then necessitates additional templates to enforce the constraint that text subsequences cannot both be different entities, something that is more naturally modeled by a CRF. Another disadvantage of this approach is that it uses *loopy belief propagation* and a voted perceptron for approximate learning and inference – ill-founded and inherently unstable algorithms which are noted by the authors to have caused convergence problems. In the *skip-chain CRFs* model, the decision of which nodes to connect is also made heuristically, and because the authors focus on named entity recognition, they chose to connect all pairs of identical capitalized words. They also utilize loopy belief propagation for approximate learning and inference.

While the technique we propose is similar mathematically and in spirit to the above approaches, it differs in some important ways. Our model is implemented by adding additional constraints into the model at inference time, and does not require the preprocessing step necessary in the two previously mentioned works. This allows for a broader class of long-distance dependencies, because we do not need to make any initial assumptions about which nodes should be connected, and is helpful when you wish to model relationships between nodes which are the same class, but may not be similar in any other way. For instance, in the CMU Seminar Announcements dataset, we can normalize all entities labeled as a *start time* and penalize the model if multiple, nonconsistent times are labeled. This type of constraint cannot be modeled in an RMN or a skip-CRF, because it requires the knowledge that both entities are given the same class label.

We also allow dependencies between multi-word phrases, and not just single words. Additionally, our model can be applied on top of a pre-existing trained sequence model. As such, our method does not require complex training procedures, and can instead leverage all of the established methods for training high accuracy sequence models. It can indeed be used in conjunction with any statistical hidden state sequence model: HMMs, CMMs, CRFs, or even heuristic models. Third, our technique employs Gibbs sampling for approximate inference, a simple and probabilistically well-founded algorithm. As a consequence of these differences, our approach is easier to understand, implement, and adapt to new applications.

9 Conclusions

We have shown that a constraint model can be effectively combined with an existing sequence model in a factored architecture to successfully impose various sorts of long distance constraints. Our model generalizes naturally to other statistical models and other tasks. In particular, it could in the future be applied to statistical parsing. Statistical context free grammars provide another example of statistical models which are restricted to limiting local structure, and which could benefit from modeling nonlocal structure.

Acknowledgements

This work was supported in part by the Advanced Researchand Development Activity (ARDA)'s Advanced Question Answeringfor Intelligence (AQUAINT) Program. Additionally, we would like to that our reviewers for their helpful comments.

References

S. Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23:597–618.

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. Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University.

R. Bunescu and R. J. Mooney. 2004. Collective information extraction with relational Markov networks. In *Proceedings of the 42nd ACL*, pages 439–446.

H. L. Chieu and H. T. Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th Coling*, pages 190–196.

R. G. Cowell, A. Philip Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

J. R. Curran and S. Clark. 2003. Language independent NER using a maximum entropy tagger. In *Proceedings of the 7th CoNLL*, pages 164–167.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393.

J. Finkel, S. Dingare, H. Nguyen, M. Nissim, and C. D. Manning. 2004. Exploiting context for biomedical entity recognition: from syntax to the web. In *Joint Workshop on Natural Language Processing in Biomedicine and Its Applications at Coling 2004*.

D. Freitag and A. McCallum. 1999. Information extraction with HMMs and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*.

D. Freitag. 1998. *Machine learning for information extraction in informal domains*. Ph.D. thesis, Carnegie Mellon University.

S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transitions on Pattern Analysis and Machine Intelligence*, 6:721–741.

M. Kim, Y. S. Han, and K. Choi. 1995. Collocation map for overcoming data sparseness. In *Proceedings of the 7th EACL*, pages 53–59.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220:671–680.

P. J. Van Laarhoven and E. H. L. Arts. 1987. *Simulated Annealing: Theory and Applications*. Reidel Publishers.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th ICML*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

T. R. Leek. 1997. Information extraction using hidden Markov models. Master's thesis, U.C. San Diego.

R. Malouf. 2002. Markov models for language-independent named entity recognition. In *Proceedings of the 6th CoNLL*, pages 187–190.

A. Mikheev, M. Moens, and C. Grover. 1999. Named entity recognition without gazetteers. In *Proceedings of the 9th EACL*, pages 1–8.

L. R. Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

C. Sutton and A. McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning and Its connections to Other Fields*.

B. Taskar, P. Abbeel, and D. Koller. 2002. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertianty in Artificial Intelligence (UAI-02)*, pages 485–494, Edmonton, Canada.

Unsupervised Learning of Field Segmentation Models for Information Extraction

Trond Grenager
Computer Science Department
Stanford University
Stanford, CA 94305
grenager@cs.stanford.edu

Dan Klein
Computer Science Division
U.C. Berkeley
Berkeley, CA 94709
klein@cs.berkeley.edu

Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305
manning@cs.stanford.edu

Abstract

The applicability of many current information extraction techniques is severely limited by the need for supervised training data. We demonstrate that for certain *field structured* extraction tasks, such as classified advertisements and bibliographic citations, small amounts of prior knowledge can be used to learn effective models in a primarily unsupervised fashion. Although hidden Markov models (HMMs) provide a suitable generative model for field structured text, general unsupervised HMM learning fails to learn useful structure in either of our domains. However, one can dramatically improve the quality of the learned structure by exploiting simple prior knowledge of the desired solutions. In both domains, we found that unsupervised methods can attain accuracies with 400 unlabeled examples comparable to those attained by supervised methods on 50 labeled examples, and that semi-supervised methods can make good use of small amounts of labeled data.

1 Introduction

Information extraction is potentially one of the most useful applications enabled by current natural language processing technology. However, unlike general tools like parsers or taggers, which generalize reasonably beyond their training domains, extraction systems must be entirely retrained for each application. As an example, consider the task of turning a set of diverse classified advertisements into a queryable database; each type of ad would require tailored training data for a supervised system. Approaches which required little or no training data would therefore provide substantial resource savings and extend the practicality of extraction systems.

The term *information extraction* was introduced in the MUC evaluations for the task of finding short pieces of relevant information within a broader text that is mainly irrelevant, and returning it in a structured form. For such "nugget extraction" tasks, the use of unsupervised learning methods is difficult and unlikely to be fully successful, in part because the nuggets of interest are determined only extrinsically by the needs of the user or task. However, the term *information extraction* was in time generalized to a related task that we distinguish as *field segmentation*. In this task, a document is regarded as a sequence of pertinent fields, and the goal is to segment the document into fields, and to label the fields. For example, bibliographic citations, such as the one in Figure 1(a), exhibit clear field structure, with fields such as *author*, *title*, and *date*. Classified advertisements, such as the one in Figure 1(b), also exhibit field structure, if less rigidly: an ad consists of descriptions of attributes of an item or offer, and a set of ads for similar items share the same attributes. In these cases, the fields present a salient, intrinsic form of linguistic structure, and it is reasonable to hope that field segmentation models could be learned in an unsupervised fashion.

In this paper, we investigate unsupervised learning of field segmentation models in two domains: bibliographic citations and classified advertisements for apartment rentals. General, unconstrained induction of HMMs using the EM algorithm fails to detect useful field structure in either domain. However, we demonstrate that small amounts of prior knowledge can be used to greatly improve the learned model. In both domains, we found that unsupervised methods can attain accuracies with 400 unlabeled examples comparable to those attained by supervised methods on 50 labeled examples, and that semi-supervised methods can make good use of small amounts of labeled data.

(a)

AUTH	*AUTH*	*AUTH*	*DATE*	*DATE*	*DATE*	*DATE*	*TTL*	*TTL*	*TTL*	*TTL*
Pearl	,	J.	(	1988	)	.	Probabilistic	Reasoning	in	Intelligent
TTL	*TTL*	*TTL*	*TTL*	*TTL*	*TTL*	*TTL*	*PUBL*	*PUBL*	*PUBL*	
Systems	:	Networks	of	Plausible	Inference	.	Morgan	Kaufmann	.	

(b)

SIZE	*SIZE*	*SIZE*	*SIZE*	*SIZE*	*FEAT*	*FEAT*	*FEAT*	*FEAT*	*FEAT*	*FEAT*
Spacious	1	Bedroom	apt	.	newly	remodeled	,	gated	,	new
FEAT	*FEAT*	*FEAT*	*FEAT*	*FEAT*	*NBRHD*	*NBRHD*	*NBRHD*	*NBRHD*	*NBRHD*	*NBRHD*
appliance	,	new	carpet	,	near	public	transportion	,	close	to
NBRHD	*NBRHD*	*NBRHD*	*RENT*	*RENT*	*RENT*	*CONTACT*				
580	freeway	,	$	500.00	Deposit	(510)655-0106				

(c)

RB	*,*	*PRP*	*VBD*	*RB*	*NNP*	*NNP*	*.*
No	,	it	was	n't	Black	Monday	.

Figure 1: Examples of three domains for HMM learning: the bibliographic citation fields in (a) and classified advertisements for apartment rentals shown in (b) exhibit field structure. Contrast these to part-of-speech tagging in (c) which does not.

2 Hidden Markov Models

Hidden Markov models (HMMs) are commonly used to represent a wide range of linguistic phenomena in text, including morphology, parts-of-speech (POS), named entity mentions, and even topic changes in discourse. An HMM consists of a set of states S, a set of observations (in our case words or tokens) W, a transition model specifying $\mathrm{P}(s_t|s_{t-1})$, the probability of transitioning from state s_{t-1} to state s_t, and an emission model specifying $\mathrm{P}(w|s)$ the probability of emitting word w while in state s. For a good tutorial on general HMM techniques, see Rabiner (1989).

For all of the unsupervised learning experiments we fit an HMM with the same number of hidden states as gold labels to an unannotated training set using EM.[1] To compute hidden state expectations efficiently, we use the Forward-Backward algorithm in the standard way. Emission models are initialized to almost-uniform probability distributions, where a small amount of noise is added to break initial symmetry. Transition model initialization varies by experiment. We run the EM algorithm to convergence. Finally, we use the Viterbi algorithm with the learned parameters to label the test data.

All baselines and experiments use the same tokenization, normalization, and smoothing techniques, which were not extensively investigated. Tokenization was performed in the style of the Penn Treebank, and tokens were normalized in various ways: numbers, dates, phone numbers, URLs, and email addresses were collapsed to dedicated tokens, and all remaining tokens were converted to lowercase. Unless otherwise noted, the emission models use simple add-λ smoothing, where λ was 0.001 for supervised techniques, and 0.2 for unsupervised techniques.

[1]EM is a greedy hill-climbing algorithm designed for this purpose, but it is not the only option; one could also use coordinate ascent methods or sampling methods.

3 Datasets and Evaluation

The bibliographic citations data is described in McCallum et al. (1999), and is distributed at *http://www.cs.umass.edu/~mccallum/*. It consists of 500 hand-annotated citations, each taken from the reference section of a different computer science research paper. The citations are annotated with 13 fields, including *author*, *title*, *date*, *journal*, and so on. The average citation has 35 tokens in 5.5 fields. We split this data, using its natural order, into a 300-document training set, a 100-document development set, and a 100-document test set.

The classified advertisements data set is novel, and consists of 8,767 classified advertisements for apartment rentals in the San Francisco Bay Area downloaded in June 2004 from the Craigslist website. It is distributed at *http://www.stanford.edu/~grenager/*. 302 of the ads have been labeled with 12 fields, including *size*, *rent*, *neighborhood*, *features*, and so on. The average ad has 119 tokens in 8.7 fields. The annotated data is divided into a 102-document training set, a 100-document development set, and a 100-document test set. The remaining 8465 documents form an unannotated training set.

In both cases, all system development and parameter tuning was performed on the development set,

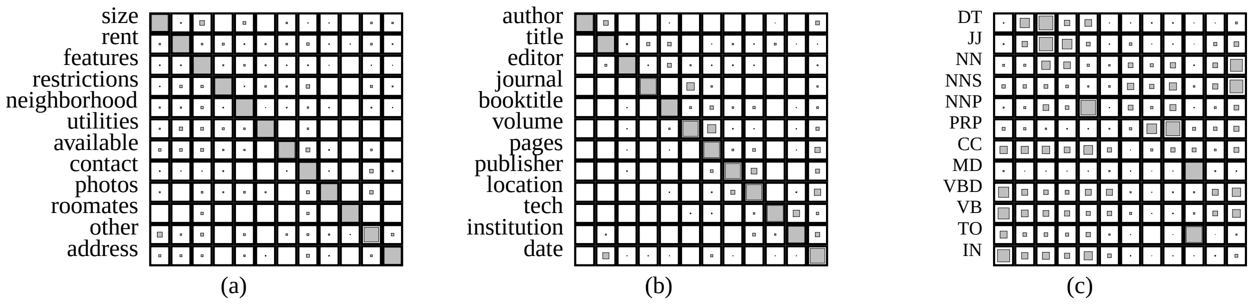


Figure 2: Matrix representations of the target transition structure in two field structured domains: (a) classified advertisements (b) bibliographic citations. Columns and rows are indexed by the same sequence of fields. Also shown is (c) a submatrix of the transition structure for a part-of-speech tagging task. In all cases the column labels are the same as the row labels.

and the test set was only used once, for running final experiments. Supervised learning experiments train on documents selected randomly from the annotated training set and test on the complete test set. Unsupervised learning experiments also test on the complete test set, but create a training set by first adding documents from the test set (without annotation), then adding documents from the annotated training set (without annotation), and finally adding documents from the unannotated training set. Thus if an unsupervised training set is larger than the test set, it fully contains the test set.

To evaluate our models, we first learn a set of model parameters, and then use the parameterized model to label the sequence of tokens in the test data with the model's hidden states. We then compare the similarity of the guessed sequence to the human-annotated sequence of gold labels, and compute accuracy on a per-token basis.[2] In evaluation of supervised methods, the model states and gold labels are the same. For models learned in a fully unsupervised fashion, we map each model state in a greedy fashion to the gold label to which it most often corresponds in the gold data. There is a worry with this kind of greedy mapping: it increasingly inflates the results as the number of hidden states grows. To keep the accuracies meaningful, all of our models have exactly the same number of hidden states as gold labels, and so the comparison is valid.

[2]This evaluation method is used by McCallum et al. (1999) but otherwise is not very standard. Compared to other evaluation methods for information extraction systems, it leads to a lower penalty for boundary errors, and allows long fields also contribute more to accuracy than short ones.

4 Unsupervised Learning

Consider the general problem of learning an HMM from an unlabeled data set. Even abstracting away from concrete search methods and objective functions, the diversity and simultaneity of linguistic structure is already worrying; in Figure 1 compare the field structure in (a) and (b) to the parts-of-speech in (c). If strong sequential correlations exist at multiple scales, any fixed search procedure will detect and model at most one of these levels of structure, not necessarily the level desired at the moment. Worse, as experience with part-of-speech and grammar learning has shown, induction systems are quite capable of producing some uninterpretable mix of various levels and kinds of structure.

Therefore, if one is to preferentially learn one kind of inherent structure over another, there must be some way of constraining the process. We could hope that field structure is the strongest effect in classified ads, while parts-of-speech is the strongest effect in newswire articles (or whatever we would try to learn parts-of-speech from). However, it is hard to imagine how one could bleach the local grammatical correlations and long-distance topical correlations from our classified ads; they are still English text with part-of-speech patterns. One approach is to vary the objective function so that the search prefers models which detect the structures which we have in mind. This is the primary way supervised methods work, with the loss function relativized to training label patterns. However, for unsupervised learning, the primary candidate for an objective function is the data likelihood, and we don't have another suggestion here. Another approach is to inject some prior knowledge into the search procedure by carefully choosing the starting point; indeed smart initialization has been critical to success in many previous unsupervised learning experiments. The central idea of this paper is that we can instead restrict the entire search domain by constraining the model class to reflect the desired structure in the data, thereby directing the search toward models of interest. We do this in several ways, which are described in the following sections.

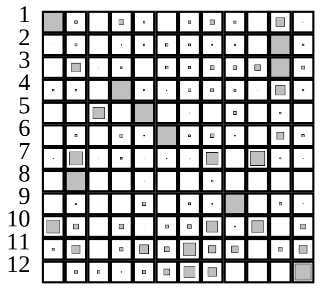
4.1 Baselines

To situate our results, we provide three different baselines (see Table 1). First is the most-frequent-field accuracy, achieved by labeling all tokens with the same single label which is then mapped to the most frequent field. This gives an accuracy of 46.4% on the advertisements data and 27.9% on the citations data. The second baseline method is to presegment the unlabeled data using a crude heuristic based on punctuation, and then to cluster the resulting segments using a simple Naïve Bayes mixture model with the Expectation-Maximization (EM) algorithm. This approach achieves an accuracy of 62.4% on the advertisements data, and 46.5% on the citations data.

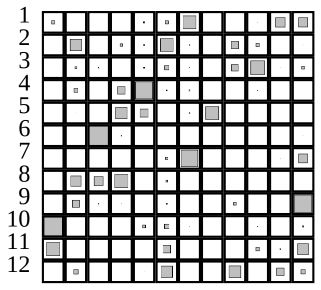
As a final baseline, we trained a supervised first-order HMM from the annotated training data using maximum likelihood estimation. With 100 training examples, supervised models achieve an accuracy of 74.4% on the advertisements data, and 72.5% on the citations data. With 300 examples, supervised methods achieve accuracies of 80.4 on the citations data. The learning curves of the supervised training experiments for different amounts of training data are shown in Figure 4. Note that other authors have achieved much higher accuracy on the the citation dataset using HMMs trained with supervision: McCallum et al. (1999) report accuracies as high as 92.9% by using more complex models and millions of words of BibTeX training data.

4.2 Unconstrained HMM Learning

From the supervised baseline above we know that there is some first-order HMM over $|S|$ states which captures the field structure we're interested in, and we would like to find such a model without supervision. As a first attempt, we try fitting an unconstrained HMM, where the transition function is initialized randomly, to the unannotated training data. Not surprisingly, the unconstrained approach leads to predictions which poorly align with the desired field segmentation: with 400 unannotated training documents, the accuracy is just 48.8% for the advertisements and 49.7% for the citations: better than the single state baseline but far from the supervised baseline. To illustrate what is (and isn't) being learned, compare typical transition models learned by this method, shown in Figure 3, to the maximum-likelihood transition models for the target annotations, shown in Figure 2. Clearly, they aren't anything like the target models: the learned classified advertisements matrix has some but not all of the desired diagonal structure, and the learned citations matrix has almost no mass on the diagonal, and appears to be modeling smaller scale structure.



(a) Classified Advertisements



(b) Citations

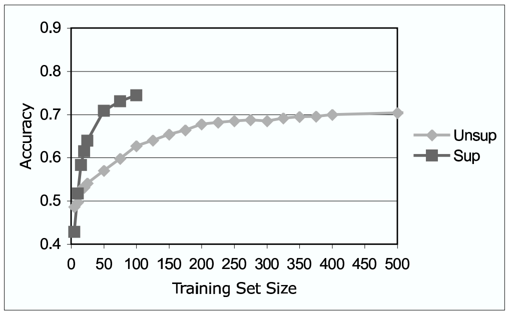
Figure 3: Matrix representations of typical transition models learned by initializing the transition model uniformly.

4.3 Diagonal Transition Models

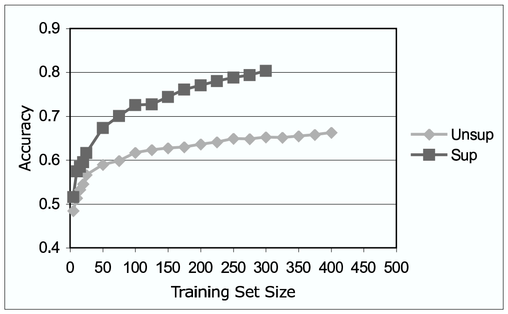
To adjust our procedure to learn larger-scale patterns, we can constrain the parametric form of the transition model to be

$$\mathrm{P}(s_t|s_{t-1}) = \begin{cases} \sigma + \frac{(1-\sigma)}{|S|} & \text{if } s_t = s_{t-1} \\ \frac{(1-\sigma)}{|S|} & \text{otherwise} \end{cases}$$

where $|S|$ is the number of states, and σ is a global free parameter specifying the self-loop probability:



(a) Classified advertisements



(b) Bibliographic citations

Figure 4: Learning curves for supervised learning and unsupervised learning with a diagonal transition matrix on (a) classified advertisements, and (b) bibliographic citations. Results are averaged over 50 runs.

the probability of a state transitioning to itself. (Note that the expected mean field length for transition functions of this form is $\frac{1}{1-\sigma}$.) This constraint provides a notable performance improvement: with 400 unannotated training documents the accuracy jumps from 48.8% to 70.0% for advertisements and from 49.7% to 66.3% for citations. The complete learning curves for models of this form are shown in Figure 4. We have tested training on more unannotated data; the slope of the learning curve is leveling out, but by training on 8000 unannotated ads, accuracy improves significantly to 72.4%. On the citations task, an accuracy of approximately 66% can be achieved either using supervised training on 50 annotated citations, or unsupervised training using 400 unannotated citations. [3]

Although σ can easily be reestimated with EM (even on a per-field basis), doing so does not yield

[3]We also tested training on 5000 additional unannotated citations collected from papers found on the Internet. Unfortunately the addition of this data didn't help accuracy. This probably results from the fact that the datasets were collected from different sources, at different times.

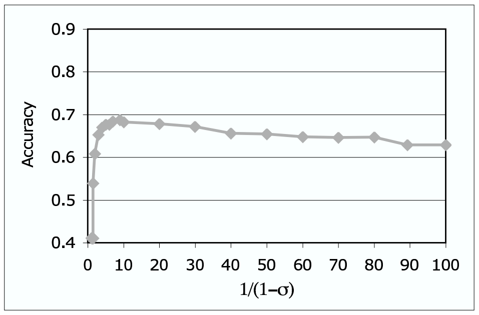


Figure 5: Unsupervised accuracy as a function of the expected mean field length $\frac{1}{1-\sigma}$ for the classified advertisements dataset. Each model was trained with 500 documents and tested on the development set. Results are averaged over 50 runs.

better models.[4] On the other hand, model accuracy is not very sensitive to the exact choice of σ, as shown in Figure 5 for the classified advertisements task (the result for the citations task has a similar shape). For the remaining experiments on the advertisements data, we use $\sigma = 0.9$, and for those on the citations data, we use $\sigma = 0.5$.

4.4 Hierarchical Mixture Emission Models

Consider the highest-probability state emissions learned by the diagonal model, shown in Figure 6(a). In addition to its characteristic content words, each state also emits punctuation and English function words devoid of content. In fact, state 3 seems to have specialized entirely in generating such tokens. This can become a problem when labeling decisions are made on the basis of the function words rather than the content words. It seems possible, then, that removing function words from the field-specific emission models could yield an improvement in labeling accuracy.

One way to incorporate this knowledge into the model is to delete stopwords, which, while perhaps not elegant, has proven quite effective in the past. A better founded way of making certain words unavailable to the model is to emit those words from all states with equal probability. This can be accomplished with the following simple hierarchical mixture emission model

$$\mathrm{P}_h(w|s) = \alpha \mathrm{P}_c(w) + (1-\alpha)\mathrm{P}(w|s)$$

where P_c is the common word distribution, and α is

[4]While it may be surprising that disallowing reestimation of the transition function is helpful here, the same has been observed in acoustic modeling (Rabiner and Juang, 1993).

State	10 Most Common Words
1	. $ no ! month deposit , pets rent available
2	, . room and with in large living kitchen -
3	. a the is and for this to , in
4	[NUM1] [NUM0] , bedroom bath / - . car garage
5	, . and a in - quiet with unit building
6	- . [TIME] [PHONE] [DAY] call [NUM8] at

(a)

State	10 Most Common Words
1	[NUM2] bedroom [NUM1] bath bedrooms large sq car ft garage
2	$ no month deposit pets lease rent available year security
3	kitchen room new , with living large floors hardwood fireplace
4	[PHONE] call please at or for [TIME] to [DAY] contact
5	san street at ave st # [NUM:DDD] francisco ca [NUM:DDDD]
6	of the yard with unit private back a building floor
Comm.	*CR* . , and - the in a / is with : of for to

(b)

Figure 6: Selected state emissions from a typical model learned from unsupervised data using the constrained transition function: (a) with a flat emission model, and (b) with a hierarchical emission model.

a new global free parameter. In such a model, before a state emits a token it flips a coin, and with probability α it allows the common word distribution to generate the token, and with probability $(1-\alpha)$ it generates the token from its state-specific emission model (see Vaithyanathan and Dom (2000) and Toutanova et al. (2001) for more on such models). We tuned α on the development set and found that a range of values work equally well. We used a value of 0.5 in the following experiments.

We ran two experiments on the advertisements data, both using the fixed transition model described in Section 4.3 and the hierarchical emission model. First, we initialized the emission model of P_c to a general-purpose list of stopwords, and did not reestimate it. This improved the average accuracy from 70.0% to 70.9%. Second, we learned the emission model of P_c using EM reestimation. Although this method did not yield a significant improvement in accuracy, it learns sensible common words: Figure 6(b) shows a typical emission model learned with this technique. Unfortunately, this technique does not yield improvements on the citations data.

4.5 Boundary Models

Another source of error concerns field boundaries. In many cases, fields are more or less correct, but the boundaries are off by a few tokens, even when punctuation or syntax make it clear to a human reader where the exact boundary should be. One way to address this is to model the fact that in this data fields often end with one of a small set of boundary tokens, such as punctuation and new lines, which are shared across states.

To accomplish this, we enriched the Markov process so that each field s is now modeled by two states, a non-final $s^- \in S^-$ and a final $s^+ \in S^+$. The transition model for final states is the same as before, but the transition model for non-final states has two new global free parameters: λ, the probability of staying within the field, and μ, the probability of transitioning to the final state given that we are staying in the field. The transition function for non-final states is then

$$\mathrm{P}(s'|s^-) = \begin{cases} (1-\mu)(\lambda + \frac{(1-\lambda)}{|S^-|}) & \text{if } s' = s^- \\ \mu(\lambda + \frac{(1-\lambda)}{|S^-|}) & \text{if } s' = s^+ \\ \frac{(1-\lambda)}{|S^-|} & \text{if } s' \in S^- \backslash s^- \\ 0 & \text{otherwise.} \end{cases}$$

Note that it can bypass the final state, and transition directly to other non-final states with probability $(1-\lambda)$, which models the fact that not all field occurrences end with a boundary token. The transition function for non-final states is then

$$\mathrm{P}(s'|s^+) = \begin{cases} \sigma + \frac{(1-\sigma)}{|S^-|} & \text{if } s' = s^- \\ \frac{(1-\sigma)}{|S^-|} & \text{if } s' \in S^- \backslash s^- \\ 0 & \text{otherwise.} \end{cases}$$

Note that this has the form of the standard diagonal function. The reason for the self-loop from the final state back to the non-final state is to allow for field internal punctuation. We tuned the free parameters on the development set, and found that $\sigma = 0.5$ and $\lambda = 0.995$ work well for the advertisements domain, and $\sigma = 0.3$ and $\lambda = 0.9$ work well for the citations domain. In all cases it works well to set $\mu = 1 - \lambda$. Emissions from non-final states are as

	Ads	Citations
Baseline	46.4	27.9
Segment and cluster	62.4	46.5
Supervised	74.4	72.5
Unsup. (learned trans)	48.8	49.7
Unsup. (diagonal trans)	70.0	66.3
+ Hierarchical (learned)	70.1	39.1
+ Hierarchical (given)	70.9	62.1
+ Boundary (learned)	70.4	64.3
+ Boundary (given)	71.9	68.2
+ Hier. + Bnd. (learned)	71.0	—
+ Hier. + Bnd. (given)	72.7	—

Table 1: Summary of results. For each experiment, we report percentage accuracy on the test set. Supervised experiments use 100 training documents, and unsupervised experiments use 400 training documents. Because unsupervised techniques are stochastic, those results are averaged over 50 runs, and differences greater than 1.0% are significant at p=0.05% or better according to the t-test. The last 6 rows are not cumulative.

before (hierarchical or not depending on the experiment), while all final states share a boundary emission model. Note that the boundary emissions are not smoothed like the field emissions.

We tested both supplying the boundary token distributions and learning them with reestimation during EM. In experiments on the advertisements data we found that learning the boundary emission model gives an insignificant raise from 70.0% to 70.4%, while specifying the list of allowed boundary tokens gives a significant increase to 71.9%. When combined with the given hierarchical emission model from the previous section, accuracy rises to 72.7%, our best unsupervised result on the advertisements data with 400 training examples. In experiments on the citations data we found that learning boundary emission model hurts accuracy, but that given the set of boundary tokens it boosts accuracy significantly: increasing it from 66.3% to 68.2%.

5 Semi-supervised Learning

So far, we have largely focused on incorporating prior knowledge in rather general and implicit ways. As a final experiment we tested the effect of adding a small amount of supervision: augmenting the large amount of unannotated data we use for unsupervised learning with a small amount of annotated data. There are many possible techniques for semi-supervised learning; we tested a particularly simple one. We treat the annotated labels as observed variables, and when computing sufficient statistics in the M-step of EM we add the observed counts from the

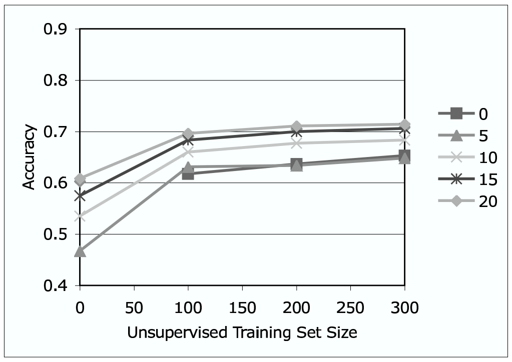


Figure 7: Learning curves for semi-supervised learning on the citations task. A separate curve is drawn for each number of annotated documents. All results are averaged over 50 runs.

annotated documents to the expected counts computed in the E-step. We estimate the transition function using maximum likelihood from the annotated documents only, and do not reestimate it. Semi-supervised results for the citations domain are shown in Figure 7. Adding 5 annotated citations yields no improvement in performance, but adding 20 annotated citations to 300 unannotated citations boosts performance greatly from 65.2% to 71.3%. We also tested the utility of this approach in the classified advertisement domain, and found that it did not improve accuracy. We believe that this is because the transition information provided by the supervised data is very useful for the citations data, which has regular transition structure, but is not as useful for the advertisements data, which does not.

6 Previous Work

A good amount of prior research can be cast as supervised learning of field segmentation models, using various model families and applied to various domains. McCallum et al. (1999) were the first to compare a number of supervised methods for learning HMMs for parsing bibliographic citations. The authors explicitly claim that the domain would be suitable for unsupervised learning, but they do not present experimental results. McCallum et al. (2000) applied supervised learning of Maximum Entropy Markov Models (MEMMs) to the domain of parsing Frequently Asked Question (FAQ) lists into their component field structure. More recently, Peng and McCallum (2004) applied supervised learning of Conditional Random Field (CRF) sequence models to the problem of parsing the head-ers of research papers.

There has also been some previous work on unsupervised learning of field segmentation models in particular domains. Pasula et al. (2002) performs limited unsupervised segmentation of bibliographic citations as a small part of a larger probabilistic model of identity uncertainty. However, their system does not explicitly learn a field segmentation model for the citations, and encodes a large amount of hand-supplied information about name forms, abbreviation schemes, and so on. More recently, Barzilay and Lee (2004) defined *content models*, which can be viewed as field segmentation models occurring at the level of discourse. They perform unsupervised learning of these models from sets of news articles which describe similar events. The *fields* in that case are the topics discussed in those articles. They consider a very different set of applications from the present work, and show that the learned topic models improve performance on two discourse-related tasks: information ordering and extractive document summarization. Most importantly, their learning method differs significantly from ours; they use a complex and special purpose algorithm, which is difficult to adapt, while we see our contribution to be a demonstration of the interplay between model family and learned structure. Because the structure of the HMMs they learn is similar to ours it seems that their system could benefit from the techniques of this paper. Finally, Blei and Moreno (2001) use an HMM augmented by an aspect model to automatically segment documents, similar in goal to the system of Hearst (1997), but using techniques more similar to the present work.

7 Conclusions

In this work, we have examined the task of learning field segmentation models using unsupervised learning. In two different domains, classified advertisements and bibliographic citations, we showed that by constraining the model class we were able to restrict the search space of EM to models of interest. We used unsupervised learning methods with 400 documents to yield field segmentation models of a similar quality to those learned using supervised learning with 50 documents. We demonstrated that further refinements of the model structure, including hierarchical mixture emission models and boundary models, produce additional increases in accuracy. Finally, we also showed that semi-supervised methods with a modest amount of labeled data can sometimes be effectively used to get similar good results, depending on the nature of the problem.

While there are enough resources for the citation task that much better numbers than ours can be and have been obtained (with more knowledge and resource intensive methods), in domains like classified ads for lost pets or used bicycles unsupervised learning may be the only practical option. In these cases, we find it heartening that the present systems do as well as they do, even without field-specific prior knowledge.

8 Acknowledgements

We would like to thank the reviewers for their consideration and insightful comments.

References

R. Barzilay and L. Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of HLT-NAACL 2004*, pages 113–120.

D. Blei and P. Moreno. 2001. Topic segmentation with an aspect hidden Markov model. In *Proceedings of the 24th SIGIR*, pages 343–348.

M. A. Hearst. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

A. McCallum, K. Nigam, J. Rennie, and K. Seymore. 1999. A machine learning approach to building domain-specific search engines. In *IJCAI-1999*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the 17th ICML*, pages 591–598. Morgan Kaufmann, San Francisco, CA.

H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. 2002. Identity uncertainty and citation matching. In *Proceedings of NIPS 2002*.

F. Peng and A. McCallum. 2004. Accurate information extraction from research papers using Conditional Random Fields. In *Proceedings of HLT-NAACL 2004*.

L. R. Rabiner and B.-H. Juang. 1993. *Fundamentals of Speech Recognition*. Prentice Hall.

L. R. Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

K. Toutanova, F. Chen, K. Popat, and T. Hofmann. 2001. Text classification in a hierarchical mixture model for small training sets. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 105–113. ACM Press.

S. Vaithyanathan and B. Dom. 2000. Model-based hierarchical clustering. In *UAI-2000*.

A Semantic Approach to IE Pattern Induction

Mark Stevenson and Mark A. Greenwood
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, UK
marks,m.greenwood@dcs.shef.ac.uk

Abstract

This paper presents a novel algorithm for the acquisition of Information Extraction patterns. The approach makes the assumption that useful patterns will have similar meanings to those already identified as relevant. Patterns are compared using a variation of the standard vector space model in which information from an ontology is used to capture semantic similarity. Evaluation shows this algorithm performs well when compared with a previously reported document-centric approach.

1 Introduction

Developing systems which can be easily adapted to new domains with the minimum of human intervention is a major challenge in Information Extraction (IE). Early IE systems were based on knowledge engineering approaches but suffered from a knowledge acquisition bottleneck. For example, Lehnert et al. (1992) reported that their system required around 1,500 person-hours of expert labour to modify for a new extraction task. One approach to this problem is to use machine learning to automatically learn the domain-specific information required to port a system (Riloff, 1996). Yangarber et al. (2000) proposed an algorithm for learning extraction patterns for a small number of examples which greatly reduced the burden on the application developer and reduced the knowledge acquisition bottleneck.

Weakly supervised algorithms, which bootstrap from a small number of examples, have the advantage of requiring only small amounts of annotated data, which is often difficult and time-consuming to produce. However, this also means that there are fewer examples of the patterns to be learned, making the learning task more challenging. Providing the learning algorithm with access to additional knowledge can compensate for the limited number of annotated examples. This paper presents a novel weakly supervised algorithm for IE pattern induction which makes use of the WordNet ontology (Fellbaum, 1998).

Extraction patterns are potentially useful for many language processing tasks, including question answering and the identification of lexical relations (such as meronomy and hyponymy). In addition, IE patterns encode the different ways in which a piece of information can be expressed in text. For example, “Acme Inc. fired Jones”, “Acme Inc. let Jones go”, and “Jones was given notice by his employers, Acme Inc.” are all ways of expressing the same fact. Consequently the generation of extraction patterns is pertinent to paraphrase identification which is central to many language processing problems.

We begin by describing the general process of pattern induction and an existing approach, based on the distribution of patterns in a corpus (Section 2). We then introduce a new algorithm which makes use of WordNet to generalise extraction patterns (Section 3) and describe an implementation (Section 4). Two evaluation regimes are described; one based on the identification of relevant documents and another which aims to identify sentences in a corpus which

Proceedings of the 43rd Annual Meeting of the ACL, pages 379–386,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

are relevant for a particular IE task (Section 5). Results on each of these evaluation regimes are then presented (Sections 6 and 7).

2 Extraction Pattern Learning

We begin by outlining the general process of learning extraction patterns, similar to one presented by (Yangarber, 2003).

1. For a given IE scenario we assume the existence of a set of documents against which the system can be trained. The documents are unannotated and may be either relevant (contain the description of an event relevant to the scenario) or irrelevant although the algorithm has no access to this information.
2. This corpus is pre-processed to generate the set of all patterns which could be used to represent sentences contained in the corpus, call this set S. The aim of the learning process is to identify the subset of S representing patterns which are relevant to the IE scenario.
3. The user provides a small set of seed patterns, S_{seed}, which are relevant to the scenario. These patterns are used to form the set of currently accepted patterns, S_{acc}, so $S_{acc} \leftarrow S_{seed}$. The remaining patterns are treated as candidates for inclusion in the accepted set, these form the set $S_{cand}(= S - S_{acc})$.
4. A function, f, is used to assign a score to each pattern in S_{cand} based on those which are currently in S_{acc}. This function assigns a real number to candidate patterns so $\forall\ c \ \epsilon\ S_{cand},\ f(c, S_{acc}) \ \mapsto\ \Re$. A set of high scoring patterns (based on absolute scores or ranks after the set of patterns has been ordered by scores) are chosen as being suitable for inclusion in the set of accepted patterns. These form the set S_{learn}.
5. The patterns in S_{learn} are added to S_{acc} and removed from S_{cand}, so $S_{acc} \leftarrow S_{acc} \cup S_{learn}$ and $S_{cand} \leftarrow S_{acc}\ -\ S_{learn}$
6. If a suitable set of patterns has been learned then stop, otherwise go to step 4

2.1 Document-centric approach

A key choice in the development of such an algorithm is step 4, the process of ranking the candidate patterns, which effectively determines which of the candidate patterns will be learned. Yangarber et al. (2000) chose an approach motivated by the assumption that documents containing a large number of patterns already identified as relevant to a particular IE scenario are likely to contain further relevant patterns. This approach, which can be viewed as being document-centric, operates by associating confidence scores with patterns and relevance scores with documents. Initially seed patterns are given a maximum confidence score of 1 and all others a 0 score. Each document is given a relevance score based on the patterns which occur within it. Candidate patterns are ranked according to the proportion of relevant and irrelevant documents in which they occur, those found in relevant documents far more than in irrelevant ones are ranked highly. After new patterns have been accepted all patterns' confidence scores are updated, based on the documents in which they occur, and documents' relevance according to the accepted patterns they contain.

This approach has been shown to successfully acquire useful extraction patterns which, when added to an IE system, improved its performance (Yangarber et al., 2000). However, it relies on an assumption about the way in which relevant patterns are distributed in a document collection and may learn patterns which tend to occur in the same documents as relevant ones whether or not they are actually relevant. For example, we could imagine an IE scenario in which relevant documents contain a piece of information which is related to, but distinct from, the information we aim to extract. If patterns expressing this information were more likely to occur in relevant documents than irrelevant ones the document-centric approach would also learn the irrelevant patterns.

Rather than focusing on the documents matched by a pattern, an alternative approach is to rank patterns according to how similar their meanings are to those which are known to be relevant. This semantic-similarity approach avoids the problem which may be present in the document-centric approach since patterns which happen to co-occur in the same documents as relevant ones but have different meanings will not be ranked highly. We now go on to describe a new algorithm which implements this approach.

3 Semantic IE Pattern Learning

For these experiments extraction patterns consist of predicate-argument structures, as proposed by Yangarber (2003). Under this scheme patterns consist of triples representing the subject, verb and object (SVO) of a clause. The first element is the "semantic" subject (or agent), for example "John" is a clausal subject in each of these sentences "John hit Bill", "Bill was hit by John", "Mary saw John hit Bill", and "John is a bully". The second element is the verb in the clause and the third the object (patient) or predicate. "Bill" is a clausal object in the first three example sentences and "bully" in the final one. When a verb is being used intransitively, the pattern for that clause is restricted to only the first pair of elements.

The filler of each pattern element can be either a lexical item or semantic category such as person name, country, currency values, numerical expressions etc. In this paper lexical items are represented in lower case and semantic categories are capitalised. For example, in the pattern `COMPANY+fired+ceo`, `fired` and `ceo` are lexical items and `COMPANY` a semantic category which could match any lexical item belonging to that type.

The algorithm described here relies on identifying patterns with similar meanings. The approach we have developed to do this is inspired by the vector space model which is commonly used in Information Retrieval (Salton and McGill, 1983) and language processing in general (Pado and Lapata, 2003). Each pattern can be represented as a set of pattern element-filler pairs. For example, the pattern `COMPANY+fired+ceo` consists of three pairs: `subject_COMPANY`, `verb_fired` and `object_ceo`. Each pair consists of either a lexical item or semantic category, and pattern element. Once an appropriate set of pairs has been established a pattern can be represented as a binary vector in which an element with value 1 denotes that the pattern contains a particular pair and 0 that it does not.

3.1 Pattern Similarity

The similarity of two pattern vectors can be compared using the measure shown in Equation 1. Here $\vec{a}$ and $\vec{b}$ are pattern vectors, $\vec{b^T}$ the transpose of $\vec{b}$ and

Patterns	Matrix labels
a. `chairman+resign`	1. `subject_chairman`
b. `ceo+quit`	2. `subject_ceo`
c. `chairman+comment`	3. `verb_resign`
	4. `verb_quit`
	5. `verb_comment`

Similarity matrix

$$\begin{vmatrix} 1 & 0.95 & 0 & 0 & 0 \\ 0.95 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.9 & 0.1 \\ 0 & 0 & 0.9 & 1 & 0.1 \\ 0 & 0 & 0.1 & 0.1 & 1 \end{vmatrix}$$

Similarity values

sim($\vec{a}$, $\vec{b}$) = 0.925
sim($\vec{a}$, $\vec{c}$) = 0.55
sim($\vec{b}$, $\vec{c}$) = 0.525

Figure 1: Similarity scores and matrix for an example vector space formed from three patterns

W a matrix that lists the similarity between each of the possible pattern element-filler pairs.

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a}W\vec{b^T}}{|\vec{a}||\vec{b}|} \quad (1)$$

The semantic similarity matrix W contains information about the similarity of each pattern element-filler pair stored as non-negative real numbers and is crucial for this measure. Assume that the set of patterns, P, consists of n element-filler pairs denoted by $p_1, p_2, ...p_n$. Each row and column of W represents one of these pairs and they are consistently labelled. So, for any i such that $1 \leq i \leq n$, row i and column i are both labelled with pair p_i. If w_{ij} is the element of W in row i and column j then the value of w_{ij} represents the similarity between the pairs p_i and p_j. Note that we assume the similarity of two element-filler pairs is symmetric, so $w_{ij} = w_{ji}$ and, consequently, W is a symmetric matrix. Pairs with different pattern elements (i.e. grammatical roles) are automatically given a similarity score of 0. Diagonal elements of W represent the self-similarity between pairs and have the greatest values.

Figure 1 shows an example using three patterns, `chairman+resign`, `ceo+quit` and `chairman+comment`. This shows how these patterns are represented as vectors and gives a sample semantic similarity matrix. It can be seen that the first pair of patterns are the most similar using the proposed measure.

The measure in Equation 1 is similar to the cosine metric, commonly used to determine the similarity of documents in the vector space model approach to Information Retrieval. However, the cosine metric will not perform well for our application since it does not take into account the similarity between elements of a vector and would assign equal similarity to each pair of patterns in the example shown in Figure 1.[1] The semantic similarity matrix in Equation 1 provides a mechanism to capture semantic similarity between lexical items which allows us to identify `chairman+resign` and `ceo+quit` as the most similar pair of patterns.

3.2 Populating the Matrix

It is important to choose appropriate values for the elements of W. We chose to make use of the research that has concentrated on computing similarity between pairs of lexical items using the WordNet hierarchy (Resnik, 1995; Jiang and Conrath, 1997; Patwardhan et al., 2003). We experimented with several of the measures which have been reported in the literature and found that the one proposed by Jiang and Conrath (1997) to be the most effective.

The similarity measure proposed by Jiang and Conrath (1997) relies on a technique developed by Resnik (1995) which assigns numerical values to each sense in the WordNet hierarchy based upon the amount of information it represents. These values are derived from corpus counts of the words in the synset, either directly or via the hyponym relation and are used to derive the *Information Content* (IC) of a synset c thus $IC(c) = -\log(\Pr(c))$. For two senses, s_1 and s_2, the *lowest common subsumer*, $lcs(s_1, s_2)$, is defined as the sense with the highest information content (most specific) which subsumes both senses in the WordNet hierarchy. Jiang and Conrath used these elements to calculate the semantic distance between a pair or words, w_1 and w_2, according to this formula (where $senses(w)$ is the set of all possible WordNet senses for word w):

$$\underset{\substack{s_1 \,\epsilon\, senses(w_1),\\ s_2 \,\epsilon\, senses(w_2)}}{ARGMAX} \; IC(s_1)+IC(s_2)-2\times IC(lcs(s_1,s_2)) \tag{2}$$

Patwardhan et al. (2003) convert this distance metric into a similarity measure by taking its multiplicative inverse. Their implementation was used in the experiments described later.

As mentioned above, the second part of a pattern element-filler pair can be either a lexical item or a semantic category, such as company. The identifiers used to denote these categories, i.e. `COMPANY`, do not appear in WordNet and so it is not possible to directly compare their similarity with other lexical items. To avoid this problem these tokens are manually mapped onto the most appropriate node in the WordNet hierarchy which is then used for similarity calculations. This mapping process is not particularly time-consuming since the number of named entity types with which a corpus is annotated is usually quite small. For example, in the experiments described in this paper just seven semantic classes were sufficient to annotate the corpus.

3.3 Learning Algorithm

This pattern similarity measure can be used to create a weakly supervised approach to pattern acquisition following the general outline provided in Section 2. Each candidate pattern is compared against the set of currently accepted patterns using the measure described in Section 3.1. We experimented with several techniques for ranking candidate patterns based on these scores, including using the best and average score, and found that the best results were obtained when each candidate pattern was ranked according to its score when compared against the centroid vector of the set of currently accepted patterns. We also experimented with several schemes for deciding which of the scored patterns to accept (a full description would be too long for this paper) resulting in a scheme where the four highest scoring patterns whose score is within 0.95 of the best pattern are accepted.

Our algorithm disregards any patterns whose corpus occurrences are below a set threshold, α, since these may be due to noise. In addition, a second

[1]The cosine metric for a pair of vectors is given by the calculation $\frac{a.b}{|a||b|}$. Substituting the matrix multiplication in the numerator of Equation 1 for the dot product of vectors $\vec{a}$ and $\vec{b}$ would give the cosine metric. Note that taking the dot product of a pair of vectors is equivalent to multiplying by the identity matrix, i.e. $\vec{a}.\vec{b} = \vec{a}I\vec{b}^T$. Under our interpretation of the similarity matrix, W, this equates to each pattern element-filler pair being identical to itself but not similar to anything else.

threshold, β, is used to determine the maximum number of documents in which a pattern can occur since these very frequent patterns are often too general to be useful for IE. Patterns which occur in more than $\beta \times C$, where C is the number of documents in the collection, are not learned. For the experiments in this paper we set α to 2 and β to 0.3.

4 Implementation

A number of pre-processing stages have to be applied to documents in order for the set of patterns to be extracted before learning can take place. Firstly, items belonging to semantic categories are identified by running the text through the named entity identifier in the GATE system (Cunningham et al., 2002). The corpus is then parsed, using a version of MINIPAR (Lin, 1999) adapted to process text marked with named entities, to produce dependency trees from which SVO-patterns are extracted. Active and passive voice is taken into account in MINIPAR's output so the sentences "COMPANY fired their C.E.O." and "The C.E.O. was fired by COMPANY" would yield the same triple, `COMPANY+fire+ceo`. The indirect object of ditransitive verbs is not extracted; these verbs are treated like transitive verbs for the purposes of this analysis.

An implementation of the algorithm described in Section 3 was completed in addition to an implementation of the document-centric algorithm described in Section 2.1. It is important to mention that this implementation is not identical to the one described by Yangarber et al. (2000). Their system makes some generalisations across pattern elements by grouping certain elements together. However, there is no difference between the expressiveness of the patterns learned by either approach and we do not believe this difference has any effect on the results of our experiments.

5 Evaluation

Various approaches have been suggested for the evaluation of automatic IE pattern acquisition. Riloff (1996) judged the precision of patterns learned by reviewing them manually. Yangarber et al. (2000) developed an indirect method which allowed automatic evaluation. In addition to learning a set of patterns, their system also notes the relevance of documents based on the current set of accepted patterns. Assuming the subset of documents relevant to a particular IE scenario is known, it is possible to use these relevance judgements to determine how accurately a given set of patterns can discriminate the relevant documents from the irrelevant. This evaluation is similar to the "text-filtering" sub-task used in the sixth Message Understanding Conference (MUC-6) (1995) in which systems were evaluated according to their ability to identify the documents relevant to the extraction task. The document filtering evaluation technique was used to allow comparison with previous studies.

Identifying the document containing relevant information can be considered as a preliminary stage of an IE task. A further step is to identify the sentences within those documents which are relevant. This "sentence filtering" task is a more fine-grained evaluation and is likely to provide more information about how well a given set of patterns is likely to perform as part of an IE system. Soderland (1999) developed a version of the MUC-6 corpus in which events are marked at the sentence level. The set of patterns learned by the algorithm after each iteration can be compared against this corpus to determine how accurately they identify the relevant sentences for this extraction task.

5.1 Evaluation Corpus

The evaluation corpus used for the experiments was compiled from the training and testing corpus used in MUC-6, where the task was to extract information about the movements of executives from newswire texts. A document is relevant if it has a filled template associated with it. 590 documents from a version of the MUC-6 evaluation corpus described by Soderland (1999) were used.

After the pre-processing stages described in Section 4, the MUC-6 corpus produced 15,407 pattern tokens from 11,294 different types. 10,512 patterns appeared just once and these were effectively discarded since our learning algorithm only considers patterns which occur at least twice (see Section 3.3).

The document-centric approach benefits from a large corpus containing a mixture of relevant and irrelevant documents. We provided this using a subset of the Reuters Corpus Volume I (Rose et al., 2002) which, like the MUC-6 corpus, consists of newswire

COMPANY+appoint+PERSON
COMPANY+elect+PERSON
COMPANY+promote+PERSON
COMPANY+name+PERSON
PERSON+resign
PERSON+depart
PERSON+quit

Table 1: Seed patterns for extraction task

texts. 3000 documents relevant to the management succession task (identified using document meta-data) and 3000 irrelevant documents were used to produce the supplementary corpus. This supplementary corpus yielded 126,942 pattern tokens and 79,473 types with 14,576 of these appearing more than once. Adding the supplementary corpus to the data set used by the document-centric approach led to an improvement of around 15% on the document filtering task and over 70% for sentence filtering. It was not used for the semantic similarity algorithm since there was no benefit.

The set of seed patterns listed in Table 1 are indicative of the management succession extraction task and were used for these experiments.

6 Results

6.1 Document Filtering

Results for both the document and sentence filtering experiments are reported in Table 2 which lists precision, recall and F-measure for each approach on both evaluations. Results from the document filtering experiment are shown on the left hand side of the table and continuous F-measure scores for the same experiment are also presented in graphical format in Figure 2. While the document-centric approach achieves the highest F-measure of either system (0.83 on the 33rd iteration compared against 0.81 after 48 iterations of the semantic similarity approach) it only outperforms the proposed approach for a few iterations. In addition the semantic similarity approach learns more quickly and does not exhibit as much of a drop in performance after it has reached its best value. Overall the semantic similarity approach was found to be significantly better than the document-centric approach ($p < 0.001$, Wilcoxon Signed Ranks Test).

Although it is an informative evaluation, the document filtering task is limited for evaluating IE pat-

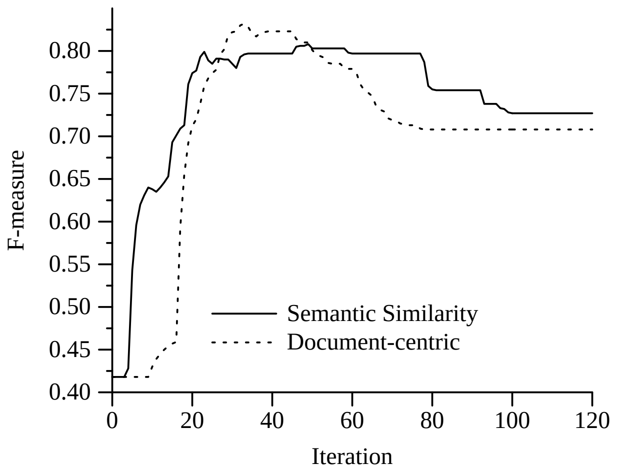


Figure 2: Evaluating document filtering.

tern learning. This evaluation indicates whether the set of patterns being learned can identify documents containing descriptions of events but does not provide any information about whether it can find those events within the documents. In addition, the set of seed patterns used for these experiments have a high precision and low recall (Table 2). We have found that the distribution of patterns and documents in the corpus means that learning virtually any pattern will help improve the F-measure. Consequently, we believe the sentence filtering evaluation to be more useful for this problem.

6.2 Sentence Filtering

Results from the sentence filtering experiment are shown in tabular format in the right hand side of Table 2[2] and graphically in Figure 3. The semantic similarity algorithm can be seen to outperform the document-centric approach. This difference is also significant ($p < 0.001$, Wilcoxon Signed Ranks Text).

The clear difference between these results shows that the semantic similarity approach can indeed identify relevant sentences while the document-centric method identifies patterns which match relevant documents, although not necessarily relevant sentences.

[2]The set of seed patterns returns a precision of 0.81 for this task. The precision is not 1 since the pattern PERSON+resign matches sentences describing historical events ("Jones resigned last year.") which were not marked as relevant in this corpus following MUC guidelines.

Number of	Document Filtering						Sentence Filtering					
	Document-centric			Semantic similarity			Document-centric			Semantic similarity		
Iterations	P	R	F	P	R	F	P	R	F	P	R	F
0	1.00	0.26	0.42	1.00	0.26	0.42	0.81	0.10	0.18	0.81	0.10	0.18
20	0.75	0.68	0.71	0.77	0.78	0.77	0.30	0.29	0.29	0.61	0.49	0.54
40	0.72	0.96	0.82	0.70	0.93	0.80	0.40	0.67	0.51	0.47	0.64	0.55
60	0.65	0.96	0.78	0.68	0.96	0.80	0.32	0.70	0.44	0.42	0.73	0.54
80	0.56	0.96	0.71	0.61	0.98	0.76	0.18	0.71	0.29	0.37	0.89	0.52
100	0.56	0.96	0.71	0.58	0.98	0.73	0.18	0.73	0.28	0.28	0.92	0.42
120	0.56	0.96	0.71	0.58	0.98	0.73	0.17	0.75	0.28	0.26	0.95	0.41

Table 2: Comparison of the different approaches over 120 iterations

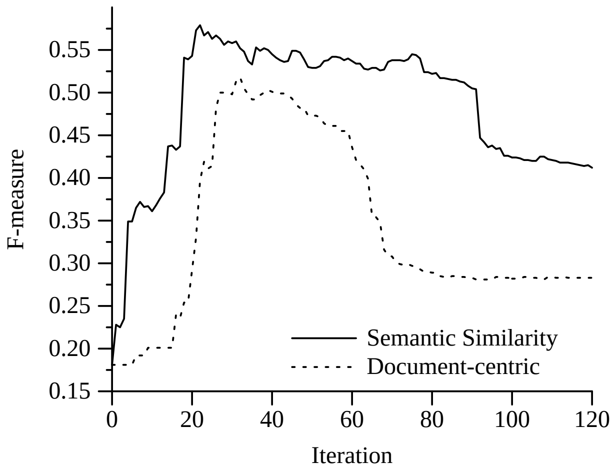


Figure 3: Evaluating sentence filtering.

The precision scores for the sentence filtering task in Table 2 show that the semantic similarity algorithm consistently learns more accurate patterns than the existing approach. At the same time it learns patterns with high recall much faster than the document-centric approach, by the 120th iteration the pattern set covers almost 95% of relevant sentences while the document-centric approach covers only 75%.

7 Discussion

The approach to IE pattern acquisition presented here is related to other techniques but uses different assumptions regarding which patterns are likely to be relevant to a particular extraction task. Evaluation has showed that the semantic generalisation approach presented here performs well when compared to a previously reported document-centric method. Differences between the two approaches are most obvious when the results of the sentence filtering task are considered and it seems that this is a more informative evaluation for this problem. The semantic similarity approach has the additional advantage of not requiring a large corpus containing a mixture of documents relevant and irrelevant to the extraction task. This corpus is unannotated, and so may not be difficult to obtain, but is nevertheless an additional requirement.

The best score recorded by the proposed algorithm on the sentence filtering task is an F-measure of 0.58 (22nd iteration). While this result is lower than those reported for IE systems based on knowledge engineering approaches these results should be placed in the context of a weakly supervised learning algorithm which could be used to complement manual approaches. These results could be improved by manual filtering the patterns identified by the algorithm.

The learning algorithm presented in Section 3 includes a mechanism for comparing two extraction patterns using information about lexical similarity derived from WordNet. This approach is not restricted to this application and could be applied to other language processing tasks such as question answering, paraphrase identification and generation or as a variant of the vector space model commonly used in Information Retrieval. In addition, Sudo et al. (2003) proposed representations for IE patterns which extends the SVO representation used here and, while they did not appear to significantly improve IE, it is expected that it will be straightforward to extend the vector space model to those pattern representations.

One of the reasons for the success of the approach described here is the appropriateness of WordNet which is constructed on paradigmatic principles, listing the words which may be substituted for one another, and is consequently an excellent resource for this application. WordNet is also a generic resource not associated with a particular domain which means the learning algorithm can make use of that knowledge to acquire patterns for a diverse range of IE tasks. This work represents a step towards truly domain-independent IE systems. Employing a weakly supervised learning algorithm removes much of the requirement for a human annotator to provide example patterns. Such approaches are often hampered by a lack of information but the additional knowledge in WordNet helps to compensate.

Acknowledgements

This work was carried out as part of the RESuLT project funded by the EPSRC (GR/T06391). Roman Yangarber provided advice on the reimplementation of the document-centric algorithm. We are also grateful for the detailed comments provided by the anonymous reviewers of this paper.

References

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: an Architecture for Development of Robust HLT. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL-02)*, pages 168–175, Philadelphia, PA.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database and some of its Applications*. MIT Press, Cambridge, MA.

J. Jiang and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research in Computational Linguistics*, Taiwan.

W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. 1992. University of Massachusetts: Description of the CIRCUS System used for MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 282–288, San Francisco, CA.

D. Lin. 1999. MINIPAR: a minimalist parser. In *Maryland Linguistics Colloquium*, University of Maryland, College Park.

MUC. 1995. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, San Mateo, CA. Morgan Kaufmann.

S. Pado and M. Lapata. 2003. Constructing semantic space models from parsed corpora. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 128–135, Sapporo, Japan.

S. Patwardhan, S. Banerjee, and T. Pedersen. 2003. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conferences on Intelligent Text Processing and Computational Linguistics*, pages 241–257, Mexico City.

P. Resnik. 1995. Using Information Content to evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 448–453, Montreal, Canada.

E. Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049, Portland, OR.

T. Rose, M. Stevenson, and M. Whitehead. 2002. The Reuters Corpus Volume 1 - from Yesterday's news to tomorrow's language resources. In *LREC-02*, pages 827–832, La Palmas, Spain.

G. Salton and M. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

S. Soderland. 1999. Learning Information Extraction Rules for Semi-structured and free text. *Machine Learning*, 31(1-3):233–272.

K. Sudo, S. Sekine, and R. Grishman. 2003. An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 224–231.

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 (COLING 2000)*, pages 940–946, Saarbrücken, Germany.

R. Yangarber. 2003. Counter-training in the discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 343–350, Sapporo, Japan.

Word Sense Disambiguation vs. Statistical Machine Translation

Marine CARPUAT `marine@cs.ust.hk`
Dekai WU[1] `dekai@cs.ust.hk`

Human Language Technology Center
HKUST
Department of Computer Science
University of Science and Technology
Clear Water Bay, Hong Kong

Abstract

We directly investigate a subject of much recent debate: do word sense disambiguation models help statistical machine translation quality? We present empirical results casting doubt on this common, but unproved, assumption. Using a state-of-the-art Chinese word sense disambiguation model to choose translation candidates for a typical IBM statistical MT system, we find that word sense disambiguation does *not* yield significantly better translation quality than the statistical machine translation system alone. Error analysis suggests several key factors behind this surprising finding, including inherent limitations of current statistical MT architectures.

1 Introduction

Word sense disambiguation or WSD, the task of determining the correct sense of a word in context, is a much studied problem area with a long and honorable history. Recent years have seen steady accuracy gains in WSD models, driven in particular by controlled evaluations such as the Senseval series of workshops. Word sense disambiguation is often assumed to be an intermediate task, which should then help higher level applications such as machine translation or information retrieval. However, WSD is usually performed and evaluated as a standalone task, and to date there have been very few efforts to integrate the learned WSD models into full statistical MT systems.

An energetically debated question at conferences over the past year is whether even the new state-of-the-art word sense disambiguation models actually have anything to offer to full statistical machine translation systems. Among WSD circles, this can sometimes elicit responses that border on implying that even asking the question is heretical. In efforts such as Senseval we tend to regard the construction of WSD models as an obviously correct, if necessarily simplified, approach that will eventually lead to essential disambiguation components within larger applications like machine translation.

There is no question that the word sense disambiguation perspective has led to numerous insights in machine translation, even of the statistical variety. It is often simply an unstated assumption that any full translation system, to achieve full performance, will sooner or later have to incorporate individual WSD components.

However, in some translation architectures and particularly in statistical machine translation (SMT), the translation engine already implicitly factors in many contextual features into lexical choice. From this standpoint, SMT models can be seen as WSD models in their own right, albeit with several major caveats.

But typical statistical machine translation models only rely on a local context to choose among lexical translation candidates, as discussed in greater detail later. It is therefore often assumed that dedicated WSD-based lexical choice models, which can incor-

[1]The authors would like to thank the Hong Kong Research Grants Council (RGC) for supporting this research in part through grants RGC6083/99E, RGC6256/00E, and DAG03/04.EG09, and several anonymous reviewers for insights and suggestions.

porate a wider variety of context features, can make better predictions than the "weaker" models implicit in statistical MT, and that these predictions will help the translation quality.

Nevertheless, this assumption has not been empirically verified, and we should not simply assume that WSD models can contribute more than what the SMT models perform. It may behoove us to take note of the sobering fact that, perhaps analogously, WSD has yet to be conclusively shown to help information retrieval systems after many years of attempts.

In this work, we propose to directly investigate whether word sense disambiguation—at least as it is typically currently formulated—is useful for statistical machine translation. We tackle a real Chinese to English translation task using a state-of-the-art supervised WSD system and a typical SMT model. We show that the unsupervised SMT model, trained on parallel data without any manual sense annotation, yields higher BLEU scores than the case where the SMT model makes use of the lexical choice predictions from the supervised WSD model, which are more expensive to create. The reasons for the surprising difficulty of improving over the translation quality of the SMT model are then discussed and analyzed.

2 Word sense disambiguation vs. statistical machine translation

We begin by examining the respective strengths and weaknesses of dedicated WSD models versus full SMT models, that could be expected to be relevant to improving lexical choice.

2.1 Features Unique to WSD

Dedicated WSD is typically cast as a classification task with a predefined sense inventory. Sense distinctions and granularity are often manually predefined, which means that they can be adapted to the task at hand, but also that the translation candidates are limited to an existing set.

To improve accuracy, dedicated WSD models typically employ features that are not limited to the local context, and that include more linguistic information than the surface form of words. This often requires several stages of preprocessing, such as part-of-speech tagging and/or parsing. (Preprocessor domain can be an issue, since WSD accuracy may suffer from domain mismatches between the data the preprocessors were trained on, and the data they are applied to.) For example, a typical dedicated WSD model might employ features as described by Yarowsky and Florian (2002) in their "feature-enhanced naive Bayes model", with position-sensitive, syntactic, and local collocational features. The feature set made available to the WSD model to predict lexical choices is therefore much richer than that used by a statistical MT model.

Also, dedicated WSD models can be supervised, which yields significantly higher accuracies than unsupervised. For the experiments described in this study we employed supervised training, exploiting the annotated corpus that was produced for the Senseval-3 evaluation.

2.2 Features Unique to SMT

Unlike lexical sample WSD models, SMT models simultaneously translate complete sentences rather than isolated target words. The lexical choices are made in a way that heavily prefers *phrasal cohesion* in the output target sentence, as scored by the language model. That is, the predictions benefit from the sentential context of the *target* language. This has the general effect of improving translation fluency.

The WSD accuracy of the SMT model depends critically on the phrasal cohesion of the target language. As we shall see, this phrasal cohesion property has strong implications for the utility of WSD.

In other work (forthcoming), we investigated the inverse question of evaluating the Chinese-to-English SMT model on word sense disambiguation performance, using standard WSD evaluation methodology and datasets from the Senseval-3 Chinese lexical sample task. We showed the accuracy of the SMT model to be significantly lower than that of all the dedicated WSD models considered, even after adding the lexical sample data to the training set for SMT to allow for a fair comparison. These results highlight the relative strength, and the potential hoped-for advantage of dedicated supervised WSD models.

3 The WSD system

The WSD system used for the experiments is based on the model that achieved the best performance, by a large margin, on the Senseval-3 Chinese lexical sample task (Carpuat *et al.*, 2004).

3.1 Classification model

The model consists of an ensemble of four voting models combined by majority vote.

The first voting model is a naive 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 voting model is a maximum entropy model (Jaynes, 1978), since Klein and Manning (2002) found that this model yielded higher accuracy than naive Bayes in a subsequent comparison of WSD performance. (Note, however, that a different subset of either Senseval-1 or Senseval-2 English lexical sample data was used for their comparison.)

The third voting model is a boosting model (Freund and Schapire, 1997), since has consistently turned in very competitive scores on related tasks such as named entity classification (Carreras *et al.*, 2002) . Specifically, an AdaBoost.MH model was used (Schapire and Singer, 2000), which is a multiclass generalization of the original boosting algorithm, with boosting on top of decision stump classifiers (i.e., decision trees of depth one).

The fourth voting model is a Kernel PCA-based model (Wu *et al.*, 2004). Kernel Principal Component Analysis (*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. (Carpuat *et al.*, 2004) showed that KPCA-based WSD models achieve close accuracies to the best individual WSD models, while having a significantly different bias.

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.

The feature set used consists of position-sensitive, syntactic, and local collocational features, as described by Yarowsky and Florian (2002).

3.2 Lexical choice mapping model

Ideally, we would like the WSD model to predict English translations given Chinese target words in context. Such a model requires Chinese training data annotated with English senses, but such data is not available. Instead, the WSD system was trained using the Senseval-3 Chinese lexical sample task data. (This is suboptimal, but reflects the difficulties that arise when considering a real translation task; we cannot assume that sense-annotated data will always be available for all language pairs.)

The Chinese lexical sample task includes 20 target words. For each word, several senses are defined using the HowNet knowledge base. There are an average of 3.95 senses per target word type, ranging from 2 to 8. Only about 37 training instances per target word are available.

For the purpose of Chinese to English translation, the WSD model should predict English translations instead of HowNet senses. Fortunately, HowNet provides English glosses. This allows us to map each HowNet sense candidate to a set of English translations, converting the monolingual Chinese WSD system into a translation lexical choice model. We further extended the mapping to include any significant translation choice considered by the SMT system but not in HowNet.

4 The SMT system

To build a representative baseline statistical machine translation system, we restricted ourselves to making use of freely available tools, since the potential contribution of WSD should be easier to see against this baseline. Note that our focus here is not on the SMT model itself; our aim is to evaluate the impact of WSD on a real Chinese to English statistical machine translation task.

Table 1: Example of the translation candidates before and after mapping for the target word “路” (*lu*)

HowNet Sense ID	HowNet glosses	HowNet glosses + improved translations
56520	distance	distance
56521	sort	sort
56524	Lu	Lu
56525, 56526, 56527, 56528	path, road, route, way	path, road, route, way, circuit, roads
56530, 56531, 56532	line, means, sequence	line, means, sequence, lines
56533, 56534	district, region	district, region

4.1 Alignment model

The alignment model was trained with GIZA++ (Och and Ney, 2003), which implements the most typical IBM and HMM alignment models. Translation quality could be improved using more advanced hybrid phrasal or tree models, but this would interfere with the questions being investigated here. The alignment model used is IBM-4, as required by our decoder. The training scheme consists of IBM-1, HMM, IBM-3 and IBM-4, following (Och and Ney, 2003).

The training corpus consists of about 1 million sentences from the United Nations Chinese-English parallel corpus from LDC. This corpus was automatically sentence-aligned, so the training data does not require as much manual annotation as for the WSD model.

4.2 Language model

The English language model is a trigram model trained on the Gigaword newswire data and on the English side of the UN and Xinhua parallel corpora. The language model is also trained using a publicly available software, the CMU-Cambridge Statistical Language Modeling Toolkit (Clarkson and Rosenfeld, 1997).

4.3 Decoding

The ISI ReWrite decoder (Germann, 2003), which implements an efficient greedy decoding algorithm, is used to translate the Chinese sentences, using the alignment model and language model previously described.

Notice that very little contextual information is available to the SMT models. Lexical choice during decoding essentially depends on the translation probabilities learned for the target word, and on the English language model scores.

5 Experimental method

5.1 Test set selection

We extracted the Chinese sentences from the NIST MTEval-04 test set that contain any of the 20 target words from the Senseval-3 Chinese lexical sample target set. For a couple of targets, no instances were available from the test set. The resulting test set contains a total of 175 sentences, which is smaller than typical MT evaluation test sets, but slightly larger than the one used for the Senseval Chinese lexical sample task.

5.2 Integrating the WSD system predictions with the SMT model

There are numerous possible ways to integrate the WSD system predictions with the SMT model. We choose two different straightforward approaches, which will help analyze the effect of the different components of the SMT system, as we will see in Section 6.5.

5.2.1 Using WSD predictions for decoding

In the first approach, we use the WSD sense predictions to constrain the set of English sense candidates considered by the decoder for each of the target words. Instead of allowing all the word translation candidates from the translation model, when we use the WSD predictions we override the translation model and force the decoder to choose the best translation from the predefined set of glosses that maps to the HowNet sense predicted by the WSD model.

Table 2: Translation quality with and without the WSD model

Translation System	**BLEU score**
SMT	0.1310
SMT + WSD for postprocessing	0.1253
SMT + WSD for decoding	0.1239
SMT + WSD for decoding with improved translation candidates	0.1232

5.2.2 Using WSD predictions for postprocessing

In the second approach, we use the WSD predictions to postprocess the output of the SMT system: in each output sentence, the translation of the target word chosen by the SMT model is directly replaced by the WSD prediction. When the WSD system predicts more than one candidate, a unique translation is randomly chosen among them. As discussed later, this approach can be used to analyze the effect of the language model on the output.

It would also be interesting to use the gold standard or correct sense of the target words instead of the WSD model predictions in these experiments. This would give an upper-bound on performance and would quantify the effect of WSD errors. However, we do not have a corpus which contains both sense annotation and multiple reference translations: the MT evaluation corpus is not annotated with the correct senses of Senseval target words, and the Senseval corpus does not include English translations of the sentences.

6 Results

6.1 Even state-of-the-art WSD does not help BLEU score

Table 2 summarizes the translation quality scores obtained with and without the WSD model. Using our WSD model to constrain the translation candidates given to the decoder hurts translation quality, as measured by the automated BLEU metric (Papineni *et al.*, 2002).

Note that we are evaluating on only difficult sentences containing the problematic target words from the lexical sample task, so BLEU scores can be expected to be on the low side.

6.2 WSD still does not help BLEU score with improved translation candidates

One could argue that the translation candidates chosen by the WSD models do not help because they are only glosses obtained from the HowNet dictionary. They consist of the root form of words only, while the SMT model can learn many more translations for each target word, including inflected forms and synonyms.

In order to avoid artificially penalizing the WSD system by limiting its translation candidates to the HowNet glosses, we expand the translation set using the bilexicon learned during translation model training. For each target word, we consider the English words that are given a high translation probability, and manually map each of these English words to the sense categories defined for the Senseval model. At decoding time, the set of translation candidates considered by the language model is therefore larger, and closer to that considered by the pure SMT system.

The results in Table 2 show that the improved translation candidates do not help BLEU score. The translation quality obtained with SMT alone is still better than when the improved WSD Model is used. The simpler approach of using WSD predictions in postprocessing yields better BLEU score than the decoding approach, but still does not outperform the SMT model.

6.3 WSD helps translation quality for very few target words

If we break down the test set and evaluate the effect of the WSD per target word, we find that for all but two of the target words WSD either hurts the BLEU score or does not help it, which shows that the decrease in BLEU is not only due to a few isolated target words for which the Senseval sense distinctions are not helpful.

6.4 The “language model effect”

Error analysis revealed some surprising effects. One particularly dismaying effect is that even in cases where the WSD model is able to predict a better target word translation than the SMT model, to use the better target word translation surprisingly often still leads to a lower BLEU score.

The phrasal coherence property can help explain this surprising effect we observed. The translation chosen by the SMT model will tend to be more likely than the WSD prediction according to the language model; otherwise, it would also have been predicted by SMT. The translation with the higher language model probability influences the translation of its neighbors, thus potentially improving BLEU score, while the WSD prediction may not have been seen occurring within phrases often enough, thereby lowering BLEU score.

For example, we observe that the WSD model sometimes correctly predicts “impact” as a better translation for “冲击” (*chongji*), where the SMT model selects “shock”. In these cases, some of the reference translations also use “impact”. However, even when the WSD model constrains the decoder to select “impact” rather than “shock”, the resulting sentence translation yields a lower BLEU score. This happens because the SMT model does not know how to use “impact” correctly (if it did, it would likely have chosen “impact” itself). Forcing the lexical choice “impact” simply causes the SMT model to generate phrases such as “against Japan for peace constitution impact” instead of “against Japan for peace constitution shocks”. This actually lowers BLEU score, because of the n-gram effects.

6.5 Using WSD predictions in postprocessing does not help BLEU score either

In the postprocessing approach, decoding is done before knowing the WSD predictions, which eliminates the “language model effect”. Even in these conditions, the SMT model alone is still the best performing system.

The postprocessing approach also outperforms the integrated decoding approach, which shows that the language model is not able to make use of the WSD predictions. One could expect that letting the decoder choose among the WSD translations also yields a better translation of the context. This is indeed the case, but for very few examples only: for instance the target word “地方” (*difang*) is better used in the integrated decoding ouput “the place of local employment” , than in the postprocessing output “the place employment situation”. Instead, the majority of cases follow the pattern illustrated by the following example where the target word is “老” (*lao*): the SMT system produces the best output (“the newly elected President will still face old problems”), the postprocessed output uses the fluent sentence with a different translation (“the newly elected President will still face outdated problems”), while the translation is not used correctly with the decoding approach (“the newly elected President will face problems still to be outdated”).

Table 3: BLEU scores per target word: WSD helps for very few target words

Target word	**SMT**	**SMT + WSD**
把握 bawo	0.1482	0.1484
包 bao	0.1891	0.1891
材料 cailiao	0.0863	0.0863
冲击 chongji	0.1396	0.1491
地方 difang	0.1233	0.1083
分子 fengzi	0.1404	0.1402
活动 huodong	0.1365	0.1465
老 lao	0.1153	0.1136
路 lu	0.1322	0.1208
起来 qilai	0.1104	0.1082
钱 qian	0.1948	0.1814
突出 tuchu	0.0975	0.0989
研究 yanjiu	0.1089	0.1089
运动 zhengdong	0.1267	0.1251
走 zhou	0.0825	0.0808

6.6 BLEU score bias

The “language model effect” highlights one of the potential weaknesses of the BLEU score. BLEU penalizes for phrasal incoherence, which in the present study means that it can sometimes sacrifice adequacy for fluency.

However, the characteristics of BLEU are by no means solely responsible for the problems with WSD that we observed. To doublecheck that n-gram effects were not unduly impacting our study, we also evaluated using BLEU-1, which gave largely similar results as the standard BLEU-4 scores reported above.

7 Related work

Most translation disambiguation tasks are defined similarly to the Senseval Multilingual lexical sample tasks. In Senseval-3, the English to Hindi translation disambigation task was defined identically to the English lexical sample task, except that the WSD models are expected to predict Hindi translations instead of WordNet senses. This differs from our approach which consists of producing the translation of complete sentences, and not only of a predefined set of target words.

Brown *et al.* (1991) proposed a WSD algorithm to disambiguate English translations of French target words based on the single most informative context feature. In a pilot study, they found that using this WSD method in their French-English SMT system helped translation quality, manually evaluated using the number of acceptable translations. However, this study is limited to the unrealistic case of words that have exactly two senses in the other language.

Most previous work has focused on the distinct problem of exploiting various bilingual resources (e.g., parallel or comparable corpora, or even MT systems) to help WSD. The goal is to achieve accurate WSD with minimum amounts of annotated data. Again, this differs from our objective which consists of using WSD to improve performance on a full machine translation task, and is measured in terms of translation quality.

For instance, Ng *et al.* (2003) showed that it is possible to use word aligned parallel corpora to train accurate supervised WSD models. The objective is different; it is not possible for us to use this method to train our WSD model without undermining the question we aim to investigate: we would need to use the SMT model to word-align the parallel sentences, which could too strongly bias the predictions of the WSD model towards those of the SMT model, instead of combining predictive information from independent sources as we aim to study here.

Other work includes Li and Li (2002) who propose a bilingual bootstrapping method to learn a translation disambiguation WSD model, and Diab (2004) who exploited large amounts of automatically generated noisy parallel data to learn WSD models in an unsupervised bootstrapping scheme.

8 Conclusion

The empirical study presented here argues that we can expect that it will be quite difficult, at the least, to use standard WSD models to obtain significant improvements to statistical MT systems, even when supervised WSD models are used. This casts significant doubt on a commonly-held, but unproven, assumption to the contrary. We have presented empirically based analysis of the reasons for this surprising finding.

We have seen that one major factor is that the statistical MT model is sufficiently accurate so that within the training domain, even the state-of-the-art dedicated WSD model is only able to improve on its lexical choice predictions in a relatively small proportion of cases.

A second major factor is that even when the dedicated WSD model makes better predictions, current statistical MT models are unable to exploit this. Under this interpretation of our results, the dependence on the language model in current SMT architectures is excessive. One could of course argue that drastically increasing the amount of training data for the language model might overcome the problems from the language model effect. Given combinatorial problems, however, there is no way at present of telling whether the amount of data needed to achieve this is realistic, particularly for translation across many different domains. On the other hand, if the SMT architecture cannot make use of WSD predictions, even when they are in fact better than the SMT's lexical choices, then perhaps some alternative model striking a different balance of adequacy and fluency is called for. Ultimately, after all, WSD is a method of compensating for sparse data. Thus it may be that the present inability of WSD models to help improve accuracy of SMT systems stems not from an inherent weakness of dedicated WSD models, but rather from limitations of present-day SMT architectures.

To further test this, our experiments could be tried on other statistical MT models. For example, the WSD model's predictions could be employed in a Bracketing ITG translation model such as Wu (1996) or Zens *et al.* (2004), or alternatively they could be incorporated as features for reranking in a maximum-entropy SMT model (Och and Ney, 2002), instead of using them to constrain the sentence translation hypotheses as done here. However, the preceding discussion argues that it is doubtful that this would produce significantly different results, since the inherent problem from the "language model effect" would largely remain, causing sentence translations that include the WSD's preferred lexical choices to be discounted. For similar reasons, we suspect our findings may also hold even for more sophisticated statistical MT models that rely heavily on n-gram language models. A more grammatically structured statistical MT model that less n-gram oriented, such as the ITG based "grammatical channel" translation model (Wu and Wong, 1998), might make more effective use of the WSD model's predictions.

References

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.

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.

Xavier Carreras, Lluís Màrques, and Lluís Padró. Named entity extraction using AdaBoost. In Dan Roth and Antal van den Bosch, editors, *Proceedings of CoNLL-2002*, pages 167–170, Taipei, Taiwan, 2002.

Philip Clarkson and Ronald Rosenfeld. Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings of Eurospeech '97*, pages 2707–2710, Rhodes, Greece, 1997.

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.

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.

Ulrich Germann. Greeedy decoding for statistical machine translation in almost linear time. In *Proceedings of HLT-NAACL-2003. Edmonton, AB, Canada*, 2003.

E.T. Jaynes. *Where do we Stand on Maximum Entropy?* MIT Press, Cambridge MA, 1978.

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.

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.

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 Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL-02, Philadelphia*, 2002.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52, 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.

Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2):135–168, 2000.

Bernhard Schölkopf, Alexander Smola, and Klaus-Rober Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5), 1998.

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, 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.

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 *Proceedings of COLING-2004*, Geneva,Switzerland, August 2004.

Word Sense Disambiguation Using Label Propagation Based Semi-Supervised Learning

Zheng-Yu Niu, Dong-Hong Ji
Institute for Infocomm Research
21 Heng Mui Keng Terrace
119613 Singapore
{zniu, dhji}@i2r.a-star.edu.sg

Chew Lim Tan
Department of Computer Science
National University of Singapore
3 Science Drive 2
117543 Singapore
tancl@comp.nus.edu.sg

Abstract

Shortage of manually sense-tagged data is an obstacle to supervised word sense disambiguation methods. In this paper we investigate a label propagation based semi-supervised learning algorithm for WSD, which combines labeled and unlabeled data in learning process to fully realize a global consistency assumption: similar examples should have similar labels. Our experimental results on benchmark corpora indicate that it consistently outperforms SVM when only very few labeled examples are available, and its performance is also better than monolingual bootstrapping, and comparable to bilingual bootstrapping.

1 Introduction

In this paper, we address the problem of word sense disambiguation (WSD), which is to assign an appropriate sense to an occurrence of a word in a given context. Many methods have been proposed to deal with this problem, including supervised learning algorithms (Leacock et al., 1998), semi-supervised learning algorithms (Yarowsky, 1995), and unsupervised learning algorithms (Schütze, 1998).

Supervised sense disambiguation has been very successful, but it requires a lot of manually sense-tagged data and can not utilize raw unannotated data that can be cheaply acquired. Fully unsupervised methods do not need the definition of senses and manually sense-tagged data, but their sense clustering results can not be directly used in many NLP tasks since there is no sense tag for each instance in clusters. Considering both the availability of a large amount of unlabelled data and direct use of word senses, semi-supervised learning methods have received great attention recently.

Semi-supervised methods for WSD are characterized in terms of exploiting unlabeled data in learning procedure with the requirement of predefined sense inventory for target words. They roughly fall into three categories according to what is used for supervision in learning process: (1) using external resources, e.g., thesaurus or lexicons, to disambiguate word senses or automatically generate sense-tagged corpus, (Lesk, 1986; Lin, 1997; McCarthy et al., 2004; Seo et al., 2004; Yarowsky, 1992), (2) exploiting the differences between mapping of words to senses in different languages by the use of bilingual corpora (e.g. parallel corpora or untagged monolingual corpora in two languages) (Brown et al., 1991; Dagan and Itai, 1994; Diab and Resnik, 2002; Li and Li, 2004; Ng et al., 2003), (3) bootstrapping sense-tagged seed examples to overcome the bottleneck of acquisition of large sense-tagged data (Hearst, 1991; Karov and Edelman, 1998; Mihalcea, 2004; Park et al., 2000; Yarowsky, 1995).

As a commonly used semi-supervised learning method for WSD, bootstrapping algorithm works by iteratively classifying unlabeled examples and adding confidently classified examples into labeled dataset using a model learned from augmented labeled dataset in previous iteration. It can be found that the affinity information among unlabeled examples is not fully explored in this bootstrapping process. Bootstrapping is based on a local consistency assumption: examples close to labeled examples within same class will have same labels, which is also the assumption underlying many supervised learning algorithms, such as kNN.

Recently a promising family of semi-supervised learning algorithms are introduced, which can effectively combine unlabeled data with labeled data in learning process by exploiting cluster structure in data (Belkin and Niyogi, 2002; Blum et al., 2004; Chapelle et al., 1991; Szummer and Jaakkola, 2001; Zhu and Ghahramani, 2002; Zhu et al., 2003). Here we investigate a label propagation based semi-supervised learning algorithm (LP algorithm) (Zhu and Ghahramani, 2002) for WSD, which works by representing labeled and unlabeled examples as vertices in a connected graph, then iteratively propagating label information from any vertex to nearby vertices through weighted edges, finally inferring the labels of unlabeled examples after this propagation process converges.

Compared with bootstrapping, LP algorithm is based on a global consistency assumption. Intuitively, if there is at least one labeled example in each cluster that consists of similar examples, then unlabeled examples will have the same labels as labeled examples in the same cluster by propagating the label information of any example to nearby examples according to their proximity.

This paper is organized as follows. First, we will formulate WSD problem in the context of semi-supervised learning in section 2. Then in section 3 we will describe LP algorithm and discuss the difference between a supervised learning algorithm (SVM), bootstrapping algorithm and LP algorithm. Section 4 will provide experimental results of LP algorithm on widely used benchmark corpora. Finally we will conclude our work and suggest possible improvement in section 5.

2 Problem Setup

Let $X = \{x_i\}_{i=1}^n$ be a set of contexts of occurrences of an ambiguous word w, where x_i represents the context of the i-th occurrence, and n is the total number of this word's occurrences. Let $S = \{s_j\}_{j=1}^c$ denote the sense tag set of w. The first l examples $x_g (1 \leq g \leq l)$ are labeled as y_g ($y_g \in S$) and other u ($l+u = n$) examples $x_h (l+1 \leq h \leq n)$ are unlabeled. The goal is to predict the sense of w in context x_h by the use of label information of x_g and similarity information among examples in X.

The cluster structure in X can be represented as a connected graph, where each vertex corresponds to an example, and the edge between any two examples x_i and x_j is weighted so that the closer the vertices in some distance measure, the larger the weight associated with this edge. The weights are defined as follows: $W_{ij} = exp(-\frac{d_{ij}^2}{\sigma^2})$ if $i \neq j$ and $W_{ii} = 0$ $(1 \leq i, j \leq n)$, where d_{ij} is the distance (ex. Euclidean distance) between x_i and x_j, and σ is used to control the weight W_{ij}.

3 Semi-supervised Learning Method

3.1 Label Propagation Algorithm

In LP algorithm (Zhu and Ghahramani, 2002), label information of any vertex in a graph is propagated to nearby vertices through weighted edges until a global stable stage is achieved. Larger edge weights allow labels to travel through easier. Thus the closer the examples, more likely they have similar labels (the global consistency assumption).

In label propagation process, the soft label of each initial labeled example is clamped in each iteration to replenish label sources from these labeled data. Thus the labeled data act like sources to push out labels through unlabeled data. With this push from labeled examples, the class boundaries will be pushed through edges with large weights and settle in gaps along edges with small weights. If the data structure fits the classification goal, then LP algorithm can use these unlabeled data to help learning classification plane.

Let $Y^0 \in N^{n \times c}$ represent initial soft labels attached to vertices, where $Y_{ij}^0 = 1$ if y_i is s_j and 0 otherwise. Let Y_L^0 be the top l rows of Y^0 and Y_U^0 be the remaining u rows. Y_L^0 is consistent with the labeling in labeled data, and the initialization of Y_U^0 can be arbitrary.

Optimally we expect that the value of W_{ij} across different classes is as small as possible and the value of W_{ij} within same class is as large as possible. This will make label propagation to stay within same class. In later experiments, we set σ as the average distance between labeled examples from different classes.

Define $n \times n$ probability transition matrix $T_{ij} = P(j \rightarrow i) = \frac{W_{ij}}{\sum_{k=1}^n W_{kj}}$, where T_{ij} is the probability to jump from example x_j to example x_i.

Compute the row-normalized matrix $\overline{T}$ by $\overline{T}_{ij} = T_{ij} / \sum_{k=1}^n T_{ik}$. This normalization is to maintain the class probability interpretation of Y.

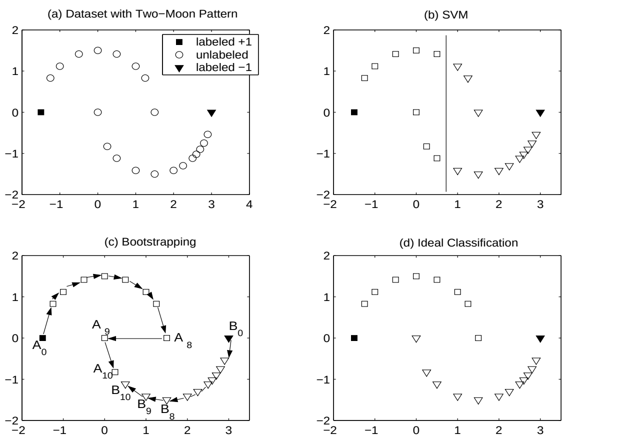


Figure 1: Classification result on two-moon pattern dataset. (a) Two-moon pattern dataset with two labeled points, (b) classification result by SVM, (c) labeling procedure of bootstrapping algorithm, (d) ideal classification.

Then LP algorithm is defined as follows:

1. Initially set t=0, where t is iteration index;

2. Propagate the label by $Y^{t+1} = \overline{T}Y^t$;

3. Clamp labeled data by replacing the top l row of Y^{t+1} with Y_L^0. Repeat from step 2 until Y^t converges;

4. Assign $x_h (l + 1 \leq h \leq n)$ with a label $s_{\hat{j}}$, where $\hat{j} = argmax_j Y_{hj}$.

This algorithm has been shown to converge to a unique solution, which is $\widehat{Y}_U = \lim_{t\to\infty} Y_U^t = (I - \overline{T}_{uu})^{-1}\overline{T}_{ul}Y_L^0$ (Zhu and Ghahramani, 2002). We can see that this solution can be obtained without iteration and the initialization of Y_U^0 is not important, since Y_U^0 does not affect the estimation of $\widehat{Y}_U$. I is $u \times u$ identity matrix. $\overline{T}_{uu}$ and $\overline{T}_{ul}$ are acquired by splitting matrix $\overline{T}$ after the l-th row and the l-th column into 4 sub-matrices.

3.2 Comparison between SVM, Bootstrapping and LP

For WSD, SVM is one of the state of the art supervised learning algorithms (Mihalcea et al., 2004), while bootstrapping is one of the state of the art semi-supervised learning algorithms (Li and Li, 2004; Yarowsky, 1995). For comparing LP with SVM and bootstrapping, let us consider a dataset with two-moon pattern shown in Figure 1(a). The upper moon consists of 9 points, while the lower moon consists of 13 points. There is only one labeled point in each moon, and other 20 points are unlabeled. The distance metric is Euclidian distance. We can see that the points in one moon should be more similar to each other than the points across the moons.

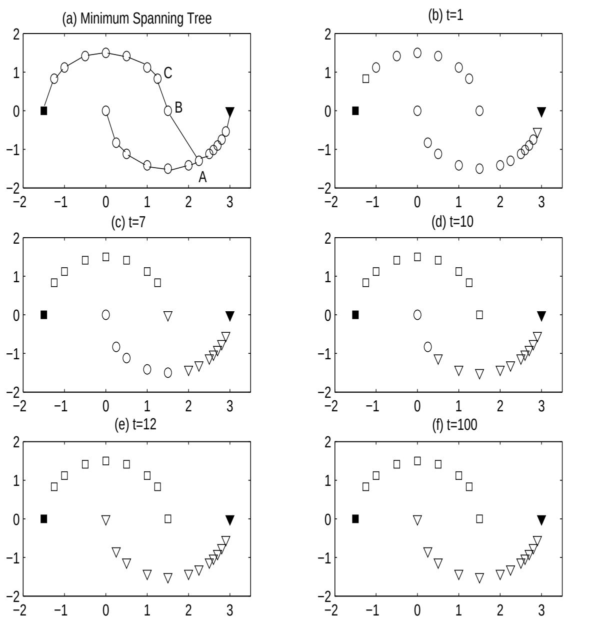


Figure 2: Classification result of LP on two-moon pattern dataset. (a) Minimum spanning tree of this dataset. The convergence process of LP algorithm with t varying from 1 to 100 is shown from (b) to (f).

Figure 1(b) shows the classification result of SVM. Vertical line denotes classification hyperplane, which has the maximum separating margin with respect to the labeled points in two classes. We can see that SVM does not work well when labeled data can not reveal the structure (two moon pattern) in each class. The reason is that the classification hyperplane was learned only from labeled data. In other words, the coherent structure (two-moon pattern) in unlabeled data was not explored when inferring class boundary.

Figure 1(c) shows bootstrapping procedure using kNN (k=1) as base classifier with user-specified parameter $b = 1$ (the number of added examples from unlabeled data into classified data for each class in each iteration). Termination condition is that the distance between labeled and unlabeled points is more than inter-class distance (the distance between A_0 and B_0). Each arrow in Figure 1(c) represents one classification operation in each iteration for each class. After eight iterations, $A_1 \sim A_8$ were tagged as $+1$, and $B_1 \sim B_8$ were tagged as -1, while $A_9 \sim A_{10}$ and $B_9 \sim B_{10}$ were still untagged. Then at the ninth iteration, A_9 was tagged as $+1$ since the label of A_9 was determined only by labeled points in kNN model: A_9 is closer to any point in $\{A_0 \sim A_8\}$ than to any point in $\{B_0 \sim B_8\}$, regardless of the intrinsic structure in data: $A_9 \sim A_{10}$ and $B_9 \sim B_{10}$ are closer to points in lower moon than to points in upper moon. In other words, bootstrapping method uses the unlabeled data under a local consistency based strategy. This is the reason that two points A_9 and A_{10} are misclassified (shown in Figure 1(c)).

From above analysis we see that both SVM and bootstrapping are based on a local consistency assumption.

Finally we ran LP on a connected graph-minimum spanning tree generated for this dataset, shown in Figure 2(a). A, B, C represent three points, and the edge $A - B$ connects the two moons. Figure 2(b)- 2(f) shows the convergence process of LP with t increasing from 1 to 100. When $t = 1$, label information of labeled data was pushed to only nearby points. After seven iteration steps ($t = 7$), point B in upper moon was misclassified as -1 since it first received label information from point A through the edge connecting two moons. After another three iteration steps (t=10), this misclassified point was retagged as $+1$. The reason of this self-correcting behavior is that with the push of label information from nearby points, the value of $Y_{B,+1}$ became higher than $Y_{B,-1}$. In other words, the weight of edge $B - C$ is larger than that of edge $B - A$, which makes it easier for $+1$ label of point C to travel to point B. Finally, when $t \geq 12$ LP converged to a fixed point, which achieved the ideal classification result.

4 Experiments and Results

4.1 Experiment Design

For empirical comparison with SVM and bootstrapping, we evaluated LP on widely used benchmark corpora - “interest”, “line” [1] and the data in English lexical sample task of SENSEVAL-3 (including all 57 English words) [2].

[1]Available at http://www.d.umn.edu/∼tpederse/data.html

[2]Available at http://www.senseval.org/senseval3

Table 1: The upper two tables summarize accuracies (averaged over 20 trials) and paired t-test results of SVM and LP on SENSEVAL-3 corpus with percentage of training set increasing from 1% to 100%. The lower table lists the official result of baseline (using most frequent sense heuristics) and top 3 systems in ELS task of SENSEVAL-3.

Percentage	SVM	LP_{cosine}	LP_{JS}
1%	24.9±2.7%	27.5±1.1%	28.1±1.1%
10%	53.4±1.1%	54.4±1.2%	54.9±1.1%
25%	62.3±0.7%	62.3±0.7%	63.3±0.9%
50%	66.6±0.5%	65.7±0.5%	66.9±0.6%
75%	68.7±0.4%	67.3±0.4%	68.7±0.3%
100%	69.7%	68.4%	70.3%

Percentage	SVM vs. LP_{cosine}		SVM vs. LP_{JS}	
	p-value	Sign.	p-value	Sign.
1%	8.7e-004	$\ll$	8.5e-005	$\ll$
10%	1.9e-006	$\ll$	1.0e-008	$\ll$
25%	9.2e-001	$\sim$	3.0e-006	$\ll$
50%	1.9e-006	$\gg$	6.2e-002	$\sim$
75%	7.4e-013	$\gg$	7.1e-001	$\sim$
100%	-	-	-	-

Systems	Baseline	htsa3	IRST-Kernels	nusels
Accuracy	55.2%	72.9%	72.6%	72.4%

We used three types of features to capture contextual information: part-of-speech of neighboring words with position information, unordered single words in topical context, and local collocations (as same as the feature set used in (Lee and Ng, 2002) except that we did not use syntactic relations). For SVM, we did not perform feature selection on SENSEVAL-3 data since feature selection deteriorates its performance (Lee and Ng, 2002). When running LP on the three datasets, we removed the features with occurrence frequency (counted in both training set and test set) less than 3 times.

We investigated two distance measures for LP: cosine similarity and Jensen-Shannon (JS) divergence (Lin, 1991).

For the three datasets, we constructed connected graphs following (Zhu et al., 2003): two instances u, v will be connected by an edge if u is among v’s k nearest neighbors, or if v is among u’s k nearest neighbors as measured by cosine or JS distance measure. For “interest” and “line” corpora, k is 10 (following (Zhu et al., 2003)), while for SENSEVAL-3 data, k is 5 since the size of dataset for each word in SENSEVAL-3 is much less than that of “interest” and “line” datasets.

4.2 Experiment 1: LP vs. SVM

In this experiment, we evaluated LP and SVM [3] on the data of English lexical sample task in SENSEVAL-3. We used l examples from training set as labeled data, and the remaining training examples and all the test examples as unlabeled data. For each labeled set size l, we performed 20 trials. In each trial, we randomly sampled l labeled examples for each word from training set. If any sense was absent from the sampled labeled set, we redid the sampling. We conducted experiments with different values of l, including $1\% \times N_{w,train}$, $10\% \times N_{w,train}$, $25\% \times N_{w,train}$, $50\% \times N_{w,train}$, $75\% \times N_{w,train}$, $100\% \times N_{w,train}$ ($N_{w,train}$ is the number of examples in training set of word w). SVM and LP were evaluated using accuracy [4] (fine-grained score) on test set of SENSEVAL-3.

We conducted paired t-test on the accuracy figures for each value of l. Paired t-test is not run when percentage$=100\%$, since there is only one paired accuracy figure. Paired t-test is usually used to estimate the difference in means between normal populations based on a set of random paired observations. $\{\ll, \gg\}$, $\{<, >\}$, and $\sim$ correspond to p-value ≤ 0.01, $(0.01, 0.05]$, and > 0.05 respectively. $\ll$ (or $\gg$) means that the performance of LP is significantly better (or significantly worse) than SVM. $<$ (or $>$) means that the performance of LP is better (or worse) than SVM. $\sim$ means that the performance of LP is almost as same as SVM.

Table 1 reports the average accuracies and paired t-test results of SVM and LP with different sizes of labled data. It also lists the official results of baseline method and top 3 systems in ELS task of SENSEVAL-3.

From Table 1, we see that with small labeled dataset (percentage of labeled data $\leq 10\%$), LP performs significantly better than SVM. When the percentage of labeled data increases from 50% to 75%, the performance of LP_{JS} and SVM become almost same, while LP_{cosine} performs significantly worse than SVM.

[3]we used linear SVM^{light}, available at http://svmlight.joachims.org/.

[4]If there are multiple sense tags for an instance in training set or test set, then only the first tag is considered as correct answer. Furthermore, if the answer of the instance in test set is "U", then this instance will be removed from test set.

Table 2: Accuracies from (Li and Li, 2004) and average accuracies of LP with $c \times b$ labeled examples on "interest" and "line" corpora. Major is a baseline method in which they always choose the most frequent sense. MB-D denotes monolingual bootstrapping with decision list as base classifier, MB-B represents monolingual bootstrapping with ensemble of Naive Bayes as base classifier, and BB is bilingual bootstrapping with ensemble of Naive Bayes as base classifier.

Ambiguous words	Accuracies from (Li and Li, 2004)			
	Major	MB-D	MB-B	BB
interest	54.6%	54.7%	69.3%	75.5%
line	53.5%	55.6%	54.1%	62.7%

Ambiguous words	Our results		
	#labeled examples	LP_{cosine}	LP_{JS}
interest	4×15=60	80.2±2.0%	79.8±2.0%
line	6×15=90	60.3±4.5%	59.4±3.9%

4.3 Experiment 2: LP vs. Bootstrapping

Li and Li (2004) used "interest" and "line" corpora as test data. For the word "interest", they used its four major senses. For comparison with their results, we took reduced "interest" corpus (constructed by retaining four major senses) and complete "line" corpus as evaluation data. In their algorithm, c is the number of senses of ambiguous word, and b ($b = 15$) is the number of examples added into classified data for each class in each iteration of bootstrapping. $c \times b$ can be considered as the size of initial labeled data in their bootstrapping algorithm. We ran LP with 20 trials on reduced "interest" corpus and complete "line" corpus. In each trial, we randomly sampled b labeled examples for each sense of "interest" or "line" as labeled data. The rest served as both unlabeled data and test data.

Table 2 summarizes the average accuracies of LP on the two corpora. It also lists the accuracies of monolingual bootstrapping algorithm (MB), bilingual bootstrapping algorithm (BB) on "interest" and "line" corpora. We can see that LP performs much better than MB-D and MB-B on both "interest" and "line" corpora, while the performance of LP is comparable to BB on these two corpora.

4.4 An Example: Word "use"

For investigating the reason for LP to outperform SVM and monolingual bootstrapping, we used the data of word "use" in English lexical sample task of SENSEVAL-3 as an example (totally 26 examples in training set and 14 examples in test set). For data

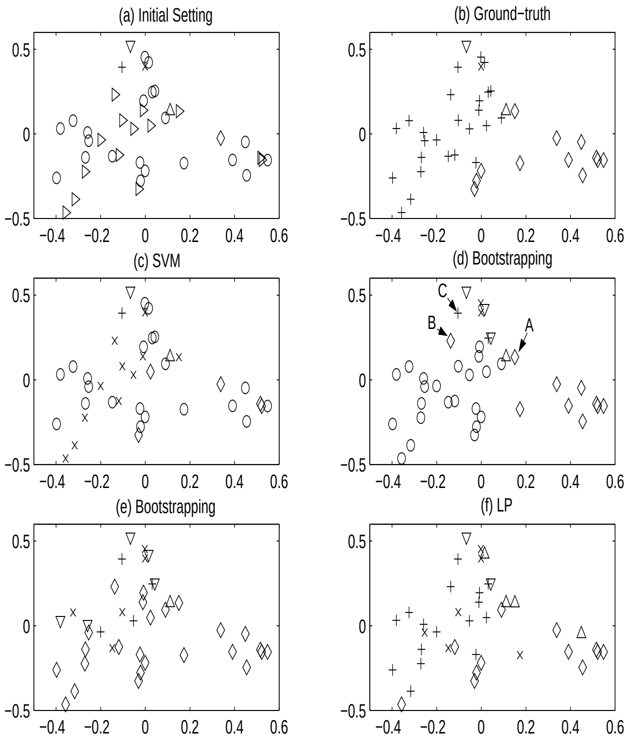


Figure 3: Comparison of sense disambiguation results between SVM, monolingual bootstrapping and LP on word "use". (a) only one labeled example for each sense of word "use" as training data before sense disambiguation (○ and ▷ denote the unlabeled examples in SENSEVAL-3 training set and test set respectively, and other five symbols ($+$, $\times$, $\triangle$, $\diamond$, and ∇) represent the labeled examples with different sense tags sampled from SENSEVAL-3 training set.), (b) ground-truth result, (c) classification result on SENSEVAL-3 test set by SVM (accuracy= $\frac{3}{14} = 21.4\%$), (d) classified data after bootstrapping, (e) classification result on SENSEVAL-3 training set and test set by 1NN (accuracy= $\frac{6}{14} = 42.9\%$), (f) classification result on SENSEVAL-3 training set and test set by LP (accuracy= $\frac{10}{14} = 71.4\%$).

visualization, we conducted unsupervised nonlinear dimensionality reduction[5] on these 40 feature vectors with 210 dimensions. Figure 3 (a) shows the dimensionality reduced vectors in two-dimensional space. We randomly sampled only one labeled example for each sense of word "use" as labeled data. The remaining data in training set and test set served as unlabeled data for bootstrapping and LP. All of these three algorithms are evaluated using accuracy on test set.

From Figure 3 (c) we can see that SVM misclassified many examples from class $+$ into class $\times$ since using only features occurring in training set can not reveal the intrinsic structure in full dataset.

For comparison, we implemented monolingual bootstrapping with kNN (k=1) as base classifier. The parameter b is set as 1. Only b unlabeled examples nearest to labeled examples and with the distance less than $d_{inter-class}$ (the minimum distance between labeled examples with different sense tags) will be added into classified data in each iteration till no such unlabeled examples can be found. Firstly we ran this monolingual bootstrapping on this dataset to augment initial labeled data. The resulting classified data is shown in Figure 3 (d). Then a 1NN model was learned on this classified data and we used this model to perform classification on the remaining unlabeled data. Figure 3 (e) reports the final classification result by this 1NN model. We can see that bootstrapping does not perform well since it is susceptible to small noise in dataset. For example, in Figure 3 (d), the unlabeled example B[6] happened to be closest to labeled example A, then 1NN model tagged example B with label $\diamond$. But the correct label of B should be $+$ as shown in Figure 3 (b). This error caused misclassification of other unlabeled examples that should have label $+$.

In LP, the label information of example C can travel to B through unlabeled data. Then example A will compete with C and other unlabeled examples around B when determining the label of B. In other words, the labels of unlabeled examples are determined not only by nearby labeled examples, but also by nearby unlabeled examples. Using this classification strategy achieves better performance than the local consistency based strategy adopted by SVM and bootstrapping.

4.5 Experiment 3: LP_{cosine} vs. LP_{JS}

Table 3 summarizes the performance comparison between LP_{cosine} and LP_{JS} on three datasets. We can see that on SENSEVAL-3 corpus, LP_{JS} per-

[5]We used $Isomap$ to perform dimensionality reduction by computing two-dimensional, 39-nearest-neighbor-preserving embedding of 210-dimensional input. $Isomap$ is available at http://isomap.stanford.edu/.

[6]In the two-dimensional space, example B is not the closest example to A. The reason is that: (1) A is not close to most of nearby examples around B, and B is not close to most of nearby examples around A; (2) we used $Isomap$ to maximally preserve the neighborhood information between any example and all other examples, which caused the loss of neighborhood information between a few example pairs for obtaining a globally optimal solution.

Table 3: Performance comparison between LP_{cosine} and LP_{JS} and the results of three model selection criteria are reported in following two tables. In the lower table, $<$ (or $>$) means that the average value of function $H(Q_{cosine})$ is lower (or higher) than $H(Q_{JS})$, and it will result in selecting cosine (or JS) as distance measure. Q_{cosine} (or Q_{JS}) represents a matrix using cosine similarity (or JS divergence). $\surd$ and $\times$ denote correct and wrong prediction results respectively, while $\circ$ means that any prediction is acceptable.

	LP_{cosine} vs. LP_{JS}	
Data	p-value	Significance
SENSEVAL-3 (1%)	1.1e-003	$\ll$
SENSEVAL-3 (10%)	8.9e-005	$\ll$
SENSEVAL-3 (25%)	9.0e-009	$\ll$
SENSEVAL-3 (50%)	3.2e-010	$\ll$
SENSEVAL-3 (75%)	7.7e-013	$\ll$
SENSEVAL-3 (100%)	-	-
interest	3.3e-002	$>$
line	8.1e-002	$\sim$

	$H(D)$	$H(W)$	$H(Y_U)$
Data	cos. vs. JS	cos. vs. JS	cos. vs. JS
SENSEVAL-3 (1%)	$>(\surd)$	$>(\surd)$	$<(\times)$
SENSEVAL-3 (10%)	$<(\times)$	$>(\surd)$	$<(\times)$
SENSEVAL-3 (25%)	$<(\times)$	$>(\surd)$	$<(\times)$
SENSEVAL-3 (50%)	$>(\surd)$	$>(\surd)$	$>(\surd)$
SENSEVAL-3 (75%)	$>(\surd)$	$>(\surd)$	$>(\surd)$
SENSEVAL-3 (100%)	$<(\circ)$	$>(\circ)$	$<(\circ)$
interest	$<(\surd)$	$>(\times)$	$<(\surd)$
line	$>(\circ)$	$>(\circ)$	$>(\circ)$

forms significantly better than LP_{cosine}, but their performance is almost comparable on "interest" and "line" corpora. This observation motivates us to automatically select a distance measure that will boost the performance of LP on a given dataset.

Cross-validation on labeled data is not feasible due to the setting of semi-supervised learning ($l \ll u$). In (Zhu and Ghahramani, 2002; Zhu et al., 2003), they suggested a label entropy criterion $H(Y_U)$ for model selection, where Y is the label matrix learned by their semi-supervised algorithms. The intuition behind their method is that good parameters should result in confident labeling. Entropy on matrix W ($H(W)$) is a commonly used measure for unsupervised feature selection (Dash and Liu, 2000), which can be considered here. Another possible criterion for model selection is to measure the entropy of $c \times c$ inter-class distance matrix D calculated on labeled data (denoted as $H(D)$), where $D_{i,j}$ represents the average distance between the i-th class and the j-th class. We will investigate three criteria, $H(D)$, $H(W)$ and $H(Y_U)$, for model selection. The distance measure can be automatically selected by minimizing the average value of function $H(D)$, $H(W)$ or $H(Y_U)$ over 20 trials.

Let Q be the $M \times N$ matrix. Function $H(Q)$ can measure the entropy of matrix Q, which is defined as (Dash and Liu, 2000):

$$S_{i,j} = \exp(-\alpha * Q_{i,j}), \quad (1)$$

$$H(Q) = -\sum_{i=1}^{M}\sum_{j=1}^{N}(S_{i,j}\log S_{i,j} + (1-S_{i,j})\log(1-S_{i,j})), \quad (2)$$

where α is positive constant. The possible value of α is $-\frac{\ln 0.5}{\bar{I}}$, where $\bar{I} = \frac{1}{MN}\sum_{i,j}Q_{i,j}$. S is introduced for normalization of matrix Q. For SENSEVAL-3 data, we calculated an overall average score of $H(Q)$ by $\sum_w \frac{N_{w,test}}{\sum_w N_{w,test}} H(Q_w)$. $N_{w,test}$ is the number of examples in test set of word w. $H(D)$, $H(W)$ and $H(Y_U)$ can be obtained by replacing Q with D, W and Y_U respectively.

Table 3 reports the automatic prediction results of these three criteria.

From Table 3, we can see that using $H(W)$ can consistently select the optimal distance measure when the performance gap between LP_{cosine} and LP_{JS} is very large (denoted by $\ll$ or $\gg$). But $H(D)$ and $H(Y_U)$ fail to find the optimal distance measure when only very few labeled examples are available (percentage of labeled data $\leq 10\%$).

$H(W)$ measures the separability of matrix W. Higher value of $H(W)$ means that distance measure decreases the separability of examples in full dataset. Then the boundary between clusters is obscured, which makes it difficult for LP to locate this boundary. Therefore higher value of $H(W)$ results in worse performance of LP.

When labeled dataset is small, the distances between classes can not be reliably estimated, which results in unreliable indication of the separability of examples in full dataset. This is the reason that $H(D)$ performs poorly on SENSEVAL-3 corpus when the percentage of labeled data is less than 25%.

For $H(Y_U)$, small labeled dataset can not reveal intrinsic structure in data, which may bias the estimation of Y_U. Then labeling confidence ($H(Y_U)$) can not properly indicate the performance of LP. This may interpret the poor performance of $H(Y_U)$ on SENSEVAL-3 data when percentage $\leq 25\%$.

5 Conclusion

In this paper we have investigated a label propagation based semi-supervised learning algorithm for WSD, which fully realizes a global consistency assumption: similar examples should have similar labels. In learning process, the labels of unlabeled examples are determined not only by nearby labeled examples, but also by nearby unlabeled examples. Compared with semi-supervised WSD methods in the first and second categories, our corpus based method does not need external resources, including WordNet, bilingual lexicon, aligned parallel corpora. Our analysis and experimental results demonstrate the potential of this cluster assumption based algorithm. It achieves better performance than SVM when only very few labeled examples are available, and its performance is also better than monolingual bootstrapping and comparable to bilingual bootstrapping. Finally we suggest an entropy based method to automatically identify a distance measure that can boost the performance of LP algorithm on a given dataset.

It has been shown that one sense per discourse property can improve the performance of bootstrapping algorithm (Li and Li, 2004; Yarowsky, 1995). This heuristics can be integrated into LP algorithm by setting weight $W_{i,j} = 1$ if the i-th and j-th instances are in the same discourse.

In the future we may extend the evaluation of LP algorithm and related cluster assumption based algorithms using more benchmark data for WSD. Another direction is to use feature clustering technique to deal with data sparseness and noisy feature problem.

Acknowledgements We would like to thank anonymous reviewers for their helpful comments. Z.Y. Niu is supported by A*STAR Graduate Scholarship.

References

Belkin, M., & Niyogi, P.. 2002. Using Manifold Structure for Partially Labeled Classification. *NIPS 15*.

Blum, A., Lafferty, J., Rwebangira, R., & Reddy, R.. 2004. Semi-Supervised Learning Using Randomized Mincuts. *ICML-2004*.

Brown P., Stephen, D.P., Vincent, D.P., & Robert, Mercer.. 1991. Word Sense Disambiguation Using Statistical Methods. *ACL-1991*.

Chapelle, O., Weston, J., & Schölkopf, B. 2002. Cluster Kernels for Semi-supervised Learning. *NIPS 15*.

Dagan, I. & Itai A.. 1994. Word Sense Disambiguation Using A Second Language Monolingual Corpus. *Computational Linguistics*, Vol. 20(4), pp. 563-596.

Dash, M., & Liu, H.. 2000. Feature Selection for Clustering. *PAKDD*(pp. 110–121).

Diab, M., & Resnik. P.. 2002. An Unsupervised Method for Word Sense Tagging Using Parallel Corpora. *ACL-2002*(pp. 255–262).

Hearst, M.. 1991. Noun Homograph Disambiguation using Local Context in Large Text Corpora. *Proceedings of the 7th Annual Conference of the UW Centre for the New OED and Text Research: Using Corpora*, 24:1, 1–41.

Karov, Y. & Edelman, S.. 1998. Similarity-Based Word Sense Disambiguation. *Computational Linguistics*, 24(1): 41-59.

Leacock, C., Miller, G.A. & Chodorow, M.. 1998. Using Corpus Statistics and WordNet Relations for Sense Identification. *Computational Linguistics*, 24:1, 147–165.

Lee, Y.K. & Ng, H.T.. 2002. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. *EMNLP-2002*, (pp. 41-48).

Lesk M.. 1986. Automated Word Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. *Proceedings of the ACM SIGDOC Conference*.

Li, H. & Li, C.. 2004. Word Translation Disambiguation Using Bilingual Bootstrapping. *Computational Linguistics*, 30(1), 1-22.

Lin, D.K.. 1997. Using Syntactic Dependency as Local Context to Resolve Word Sense Ambiguity. *ACL-1997*.

Lin, J. 1991. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information Theory*, 37:1, 145–150.

McCarthy, D., Koeling, R., Weeds, J., & Carroll, J.. 2004. Finding Predominant Word Senses in Untagged Text. *ACL-2004*.

Mihalcea R.. 2004. Co-training and Self-training for Word Sense Disambiguation. *CoNLL-2004*.

Mihalcea R., Chklovski, T., & Kilgariff, A.. 2004. The SENSEVAL-3 English Lexical Sample Task. *SENSEVAL-2004*.

Ng, H.T., Wang, B., & Chan, Y.S.. 2003. Exploiting Parallel Texts for Word Sense Disambiguation: An Empirical Study. *ACL-2003*, pp. 455-462.

Park, S.B., Zhang, B.T., & Kim, Y.T.. 2000. Word Sense Disambiguation by Learning from Unlabeled Data. *ACL-2000*.

Schütze, H.. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24:1, 97–123.

Seo, H.C., Chung, H.J., Rim, H.C., Myaeng. S.H., & Kim, S.H.. 2004. Unsupervised Word Sense Disambiguation Using WordNet Relatives. *Computer, Speech and Language*, 18:3, 253–273.

Szummer, M., & Jaakkola, T.. 2001. Partially Labeled Classification with Markov Random Walks. *NIPS 14*.

Yarowsky, D.. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. *ACL-1995*, pp. 189-196.

Yarowsky, D.. 1992. Word Sense Disambiguation Using Statistical Models of Roget's Categories Trained on Large Corpora. *COLING-1992*, pp. 454-460.

Zhu, X. & Ghahramani, Z.. 2002. Learning from Labeled and Unlabeled Data with Label Propagation. *CMU CALD tech report CMU-CALD-02-107*.

Zhu, X., Ghahramani, Z., & Lafferty, J.. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. *ICML-2003*.

Domain Kernels for Word Sense Disambiguation

Alfio Gliozzo and **Claudio Giuliano** and **Carlo Strapparava**
ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica
I-38050, Trento, ITALY
{gliozzo,giuliano,strappa}@itc.it

Abstract

In this paper we present a supervised Word Sense Disambiguation methodology, that exploits kernel methods to model sense distinctions. In particular a combination of kernel functions is adopted to estimate independently both *syntagmatic* and *domain* similarity. We defined a kernel function, namely the Domain Kernel, that allowed us to plug "external knowledge" into the supervised learning process. External knowledge is acquired from unlabeled data in a totally unsupervised way, and it is represented by means of Domain Models. We evaluated our methodology on several lexical sample tasks in different languages, outperforming significantly the state-of-the-art for each of them, while reducing the amount of labeled training data required for learning.

1 Introduction

The main limitation of many supervised approaches for Natural Language Processing (NLP) is the lack of available annotated training data. This problem is known as the Knowledge Acquisition Bottleneck.

To reach high accuracy, state-of-the-art systems for Word Sense Disambiguation (WSD) are designed according to a supervised learning framework, in which the disambiguation of each word in the lexicon is performed by constructing a different classifier. A large set of sense tagged examples is then required to train each classifier. This methodology is called *word expert* approach (Small, 1980; Yarowsky and Florian, 2002). However this is clearly unfeasible for *all-words* WSD tasks, in which all the words of an open text should be disambiguated.

On the other hand, the word expert approach works very well for *lexical sample* WSD tasks (i.e. tasks in which it is required to disambiguate only those words for which enough training data is provided). As the original rationale of the lexical sample tasks was to define a clear experimental settings to enhance the comprehension of WSD, they should be considered as *preceding* exercises to all-words tasks. However this is not the actual case. Algorithms designed for lexical sample WSD are often based on pure supervision and hence "data hungry".

We think that lexical sample WSD should regain its original *explorative* role and possibly use a minimal amount of training data, exploiting instead external knowledge acquired in an unsupervised way to reach the actual state-of-the-art performance.

By the way, minimal supervision is the basis of state-of-the-art systems for all-words tasks (e.g. (Mihalcea and Faruque, 2004; Decadt et al., 2004)), that are trained on small sense tagged corpora (e.g. SemCor), in which few examples for a subset of the ambiguous words in the lexicon can be found. Thus improving the performance of WSD systems with few learning examples is a fundamental step towards the direction of designing a WSD system that works well on real texts.

In addition, it is a common opinion that the performance of state-of-the-art WSD systems is not satisfactory from an applicative point of view yet.

To achieve these goals we identified two promising research directions:

1. Modeling independently domain and syntagmatic aspects of sense distinction, to improve the feature representation of sense tagged examples (Gliozzo et al., 2004).

2. Leveraging external knowledge acquired from unlabeled corpora.

The first direction is motivated by the linguistic assumption that syntagmatic and domain (associative) relations are both crucial to represent sense distictions, while they are basically originated by very different phenomena. Syntagmatic relations hold among words that are typically located close to each other in the same sentence in a given temporal order, while domain relations hold among words that are typically used in the same semantic domain (i.e. in texts having similar topics (Gliozzo et al., 2004)). Their different nature suggests to adopt different learning strategies to detect them.

Regarding the second direction, external knowledge would be required to help WSD algorithms to better generalize over the data available for training. On the other hand, most of the state-of-the-art supervised approaches to WSD are still completely based on "internal" information only (i.e. the only information available to the training algorithm is the set of manually annotated examples). For example, in the Senseval-3 evaluation exercise (Mihalcea and Edmonds, 2004) many lexical sample tasks were provided, beyond the usual labeled training data, with a large set of unlabeled data. However, at our knowledge, none of the participants exploited this unlabeled material. Exploring this direction is the main focus of this paper. In particular we acquire a Domain Model (DM) for the lexicon (i.e. a lexical resource representing domain associations among terms), and we exploit this information inside our supervised WSD algorithm. DMs can be automatically induced from unlabeled corpora, allowing the portability of the methodology among languages.

We identified kernel methods as a viable framework in which to implement the assumptions above (Strapparava et al., 2004).

Exploiting the properties of kernels, we have defined independently a set of domain and syntagmatic kernels and we combined them in order to define a complete kernel for WSD. The domain kernels estimate the (domain) similarity (Magnini et al., 2002) among contexts, while the syntagmatic kernels evaluate the similarity among collocations.

We will demonstrate that using DMs induced from unlabeled corpora is a feasible strategy to increase the generalization capability of the WSD algorithm. Our system far outperforms the state-of-the-art systems in all the tasks in which it has been tested. Moreover, a comparative analysis of the learning curves shows that the use of DMs allows us to remarkably reduce the amount of sense-tagged examples, opening new scenarios to develop systems for all-words tasks with minimal supervision.

The paper is structured as follows. Section 2 introduces the notion of Domain Model. In particular an automatic acquisition technique based on Latent Semantic Analysis (LSA) is described. In Section 3 we present a WSD system based on a combination of kernels. In particular we define a Domain Kernel (see Section 3.1) and a Syntagmatic Kernel (see Section 3.2), to model separately syntagmatic and domain aspects. In Section 4 our WSD system is evaluated in the Senseval-3 English, Italian, Spanish and Catalan lexical sample tasks.

2 Domain Models

The simplest methodology to estimate the similarity among the topics of two texts is to represent them by means of vectors in the Vector Space Model (VSM), and to exploit the cosine similarity. More formally, let $C = \{t_1, t_2, \ldots, t_n\}$ be a corpus, let $V = \{w_1, w_2, \ldots, w_k\}$ be its vocabulary, let $\mathbf{T}$ be the $k \times n$ term-by-document matrix representing C, such that $\mathbf{t_{i,j}}$ is the frequency of word w_i into the text t_j. The VSM is a k-dimensional space $\mathbb{R}^k$, in which the text $t_j \in C$ is represented by means of the vector $\vec{t_j}$ such that the i^{th} component of $\vec{t_j}$ is $\mathbf{t_{i,j}}$. The similarity among two texts in the VSM is estimated by computing the cosine among them.

However this approach does not deal well with lexical variability and ambiguity. For example the two sentences "*he is affected by AIDS*" and "*HIV is a virus*" do not have any words in common. In the

VSM their similarity is zero because they have orthogonal vectors, even if the concepts they express are very closely related. On the other hand, the similarity between the two sentences "*the laptop has been infected by a* ***virus***" and "*HIV is a* ***virus***" would turn out very high, due to the ambiguity of the word `virus`.

To overcome this problem we introduce the notion of *Domain Model* (DM), and we show how to use it in order to define a *domain VSM* in which texts and terms are represented in a uniform way.

A DM is composed by soft clusters of terms. Each cluster represents a semantic domain, i.e. a set of terms that often co-occur in texts having similar topics. A DM is represented by a $k \times k'$ rectangular matrix $\mathbf{D}$, containing the degree of association among terms and domains, as illustrated in Table 1.

	MEDICINE	COMPUTER_SCIENCE
HIV	1	0
AIDS	1	0
virus	0.5	0.5
laptop	0	1

Table 1: Example of Domain Matrix

DMs can be used to describe lexical ambiguity and variability. Lexical ambiguity is represented by associating one term to more than one domain, while variability is represented by associating different terms to the same domain. For example the term `virus` is associated to both the domain COMPUTER_SCIENCE and the domain MEDICINE (ambiguity) while the domain MEDICINE is associated to both the terms `AIDS` and `HIV` (variability).

More formally, let $\mathcal{D} = \{D_1, D_2, ..., D_{k'}\}$ be a set of domains, such that $k' \ll k$. A DM is fully defined by a $k \times k'$ *domain matrix* $\mathbf{D}$ representing in each cell $\mathbf{d_{i,z}}$ the *domain relevance* of term w_i with respect to the domain D_z. The domain matrix $\mathbf{D}$ is used to define a function $\mathcal{D} : \mathbb{R}^k \rightarrow \mathbb{R}^{k'}$, that maps the vectors $\vec{t_j}$ expressed into the classical VSM, into the vectors $\vec{t'_j}$ in the domain VSM. $\mathcal{D}$ is defined by[1]

$$\mathcal{D}(\vec{t_j}) = \vec{t_j}(\mathbf{I^{IDF}D}) = \vec{t'_j} \qquad (1)$$

where $\mathbf{I^{IDF}}$ is a $k \times k$ diagonal matrix such that $i^{IDF}_{i,i} = IDF(w_i)$, $\vec{t_j}$ is represented as a row vector, and $IDF(w_i)$ is the *Inverse Document Frequency* of w_i.

Vectors in the domain VSM are called Domain Vectors (DVs). DVs for texts are estimated by exploiting the formula 1, while the DV $\vec{w'_i}$, corresponding to the word $w_i \in V$ is the i^{th} row of the domain matrix $\mathbf{D}$. To be a valid domain matrix such vectors should be normalized (i,e. $\langle \vec{w'_i}, \vec{w'_i} \rangle = 1$).

In the Domain VSM the similarity among DVs is estimated by taking into account second order relations among terms. For example the similarity of the two sentences "*He is affected by AIDS*" and "*HIV is a virus*" is very high, because the terms `AIDS`, `HIV` and `virus` are highly associated to the domain MEDICINE.

A DM can be estimated from hand made lexical resources such as WORDNET DOMAINS (Magnini and Cavaglià, 2000), or by performing a term clustering process on a large corpus. We think that the second methodology is more attractive, because it allows us to automatically acquire DMs for different languages.

In this work we propose the use of Latent Semantic Analysis (LSA) to induce DMs from corpora. LSA is an unsupervised technique for estimating the similarity among texts and terms in a corpus. LSA is performed by means of a Singular Value Decomposition (SVD) of the term-by-document matrix $\mathbf{T}$ describing the corpus. The SVD algorithm can be exploited to acquire a domain matrix $\mathbf{D}$ from a large corpus C in a totally unsupervised way. SVD decomposes the term-by-document matrix $\mathbf{T}$ into three matrixes $\mathbf{T} \simeq \mathbf{V\Sigma_{k'}U}^T$ where $\mathbf{\Sigma_{k'}}$ is the diagonal $k \times k$ matrix containing the highest $k' \ll k$ eigenvalues of $\mathbf{T}$, and all the remaining elements set to 0. The parameter k' is the dimensionality of the Domain VSM and can be fixed in advance[2]. Under this setting we define the domain matrix $\mathbf{D_{LSA}}$ as

$$\mathbf{D_{LSA}} = \mathbf{I^N V}\sqrt{\mathbf{\Sigma_{k'}}} \qquad (2)$$

where $\mathbf{I^N}$ is a diagonal matrix such that $\mathbf{i^N_{i,i}} = \frac{1}{\sqrt{\langle \vec{w'_i}, \vec{w'_i} \rangle}}$, $\vec{w'_i}$ is the i^{th} row of the matrix $\mathbf{V}\sqrt{\mathbf{\Sigma_{k'}}}$.[3]

[1]In (Wong et al., 1985) the formula 1 is used to define a Generalized Vector Space Model, of which the Domain VSM is a particular instance.

[2]It is not clear how to choose the right dimensionality. In our experiments we used 50 dimensions.

[3]When $\mathbf{D_{LSA}}$ is substituted in Equation 1 the Domain VSM

3 Kernel Methods for WSD

In the introduction we discussed two promising directions for improving the performance of a supervised disambiguation system. In this section we show how these requirements can be efficiently implemented in a natural and elegant way by using kernel methods.

The basic idea behind kernel methods is to embed the data into a suitable feature space $\mathcal{F}$ via a mapping function $\phi : \mathcal{X} \rightarrow \mathcal{F}$, and then use a linear algorithm for discovering nonlinear patterns. Instead of using the explicit mapping ϕ, we can use a kernel function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, that corresponds to the inner product in a feature space which is, in general, different from the input space.

Kernel methods allow us to build a modular system, as the kernel function acts as an interface between the data and the learning algorithm. Thus the kernel function becomes the only domain specific module of the system, while the learning algorithm is a general purpose component. Potentially any kernel function can work with any kernel-based algorithm. In our system we use Support Vector Machines (Cristianini and Shawe-Taylor, 2000).

Exploiting the properties of the kernel functions, it is possible to define the kernel combination schema as

$$K_C(x_i, x_j) = \sum_{l=1}^{n} \frac{K_l(x_i, x_j)}{\sqrt{K_l(x_j, x_j) K_l(x_i, x_i)}} \tag{3}$$

Our WSD system is then defined as combination of n basic kernels. Each kernel adds some additional dimensions to the feature space. In particular, we have defined two families of kernels: *Domain* and *Syntagmatic* kernels. The former is composed by both the Domain Kernel (K_D) and the Bag-of-Words kernel (K_{BoW}), that captures domain aspects (see Section 3.1). The latter captures the syntagmatic aspects of sense distinction and it is composed by two kernels: the collocation kernel (K_{Coll}) and the Part of Speech kernel (K_{PoS}) (see Section 3.2). The WSD kernels (K'_{WSD} and K_{WSD}) are then defined by combining them (see Section 3.3).

is equivalent to a Latent Semantic Space (Deerwester et al., 1990). The only difference in our formulation is that the vectors representing the terms in the Domain VSM are normalized by the matrix $\mathbf{I^N}$, and then rescaled, according to their IDF value, by matrix $\mathbf{I^{IDF}}$. Note the analogy with the *tf idf* term weighting schema (Salton and McGill, 1983), widely adopted in Information Retrieval.

3.1 Domain Kernels

In (Magnini et al., 2002), it has been claimed 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.

This assumption can be modeled by defining a kernel that estimates the domain similarity among the contexts of the words to be disambiguated, namely the *Domain Kernel*. The Domain Kernel estimates the similarity among the topics (domains) of two texts, so to capture domain aspects of sense distinction. It is a variation of the Latent Semantic Kernel (Shawe-Taylor and Cristianini, 2004), in which a DM (see Section 2) is exploited to define an explicit mapping $\mathcal{D} : \mathbb{R}^k \rightarrow \mathbb{R}^{k'}$ from the classical VSM into the Domain VSM. The Domain Kernel is defined by

$$K_D(t_i, t_j) = \frac{\langle \mathcal{D}(t_i), \mathcal{D}(t_j) \rangle}{\sqrt{\langle \mathcal{D}(t_i), \mathcal{D}(t_j) \rangle \langle \mathcal{D}(t_i), \mathcal{D}(t_j) \rangle}} \tag{4}$$

where $\mathcal{D}$ is the Domain Mapping defined in equation 1. Thus the Domain Kernel requires a Domain Matrix $\mathbf{D}$. For our experiments we acquire the matrix $\mathbf{D_{LSA}}$, described in equation 2, from a generic collection of unlabeled documents, as explained in Section 2.

A more traditional approach to detect topic (domain) similarity is to extract Bag-of-Words (BoW) features from a large window of text around the word to be disambiguated. The BoW kernel, denoted by K_{BoW}, is a particular case of the Domain Kernel, in which $\mathbf{D} = \mathbf{I}$, and $\mathbf{I}$ is the identity matrix. The BoW kernel does not require a DM, then it can be applied to the "strictly" supervised settings, in which an external knowledge source is not provided.

3.2 Syntagmatic kernels

Kernel functions are not restricted to operate on vectorial objects $\vec{x} \in \mathbb{R}^k$. In principle kernels can be defined for any kind of object representation, as for example sequences and trees. As stated in Section 1, syntagmatic relations hold among words collocated in a particular temporal order, thus they can be modeled by analyzing sequences of words.

We identified the string kernel (or word sequence kernel) (Shawe-Taylor and Cristianini, 2004) as a valid instrument to model our assumptions. The string kernel 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 gaps they contain (gap-weighted subsequence kernel).

Formally, let V be the vocabulary, the feature space associated with the gap-weighted subsequence kernel of length n is indexed by a set I of subsequences over V of length n. The (explicit) mapping function is defined by

$$\phi_u^n(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{l(\mathbf{i})}, u \in V^n \tag{5}$$

where $u = s(\mathbf{i})$ is a subsequence of s in the positions given by the tuple $\mathbf{i}$, $l(\mathbf{i})$ is the length spanned by u, and $\lambda \in]0, 1]$ is the decay factor used to penalize non-contiguous subsequences.

The associate gap-weighted subsequence kernel is defined by

$$k^n(s_i, s_j) = \langle \phi^n(s_i), \phi^n(s_j) \rangle = \sum_{u \in V^n} \phi^n(s_i)\phi^n(s_j) \tag{6}$$

We modified the generic definition of the string kernel in order to make it able to recognize collocations in a local window of the word to be disambiguated. In particular we defined two Syntagmatic kernels: the *n-gram* Collocation Kernel and the *n-gram* PoS Kernel. The *n-gram* Collocation kernel K_{Coll}^n is defined as a gap-weighted subsequence kernel applied to sequences of lemmata around the word l_0 to be disambiguated (i.e. l_{-3}, l_{-2}, l_{-1}, l_0, l_{+1}, l_{+2}, l_{+3}). This formulation allows us to estimate the number of common (sparse) subsequences of lemmata (i.e. collocations) between two examples, in order to capture syntagmatic similarity. In analogy we defined the PoS kernel K_{PoS}^n, by setting s to the sequence of PoSs p_{-3}, p_{-2}, p_{-1}, p_0, p_{+1}, p_{+2}, p_{+3}, where p_0 is the PoS of the word to be disambiguated.

The definition of the gap-weighted subsequence kernel, provided by equation 6, depends on the parameter n, that represents the length of the subsequences analyzed when estimating the similarity among sequences. For example, K_{Coll}^2 allows us to represent the bigrams around the word to be disambiguated in a more flexible way (i.e. bigrams can be sparse). In WSD, typical features are bigrams and trigrams of lemmata and PoSs around the word to be disambiguated, then we defined the Collocation Kernel and the PoS Kernel respectively by equations 7 and 8[4].

$$K_{Coll}(s_i, s_j) = \sum_{l=1}^{p} K_{Coll}^l(s_i, s_j) \tag{7}$$

$$K_{PoS}(s_i, s_j) = \sum_{l=1}^{p} K_{PoS}^l(s_i, s_j) \tag{8}$$

3.3 WSD kernels

In order to show the impact of using Domain Models in the supervised learning process, we defined two WSD kernels, by applying the kernel combination schema described by equation 3. Thus the following WSD kernels are fully specified by the list of the kernels that compose them.

$\mathbf{K_{wsd}}$ composed by K_{Coll}, K_{PoS} and K_{BoW}

$\mathbf{K'_{wsd}}$ composed by K_{Coll}, K_{PoS}, K_{BoW} and K_D

The only difference between the two systems is that K'_{wsd} uses Domain Kernel K_D. K'_{wsd} exploits external knowledge, in contrast to K_{wsd}, whose only available information is the labeled training data.

4 Evaluation and Discussion

In this section we present the performance of our kernel-based algorithms for WSD. The objectives of these experiments are:

- to study the combination of different kernels,
- to understand the benefits of plugging external information using domain models,
- to verify the portability of our methodology among different languages.

[4]The parameters p and λ are optimized by cross-validation. The best results are obtained setting $p = 2$, $\lambda = 0.5$ for K_{Coll} and $\lambda \to 0$ for K_{PoS}.

4.1 WSD tasks

We conducted the experiments on four lexical sample tasks (English, Catalan, Italian and Spanish) of the Senseval-3 competition (Mihalcea and Edmonds, 2004). Table 2 describes the tasks by reporting the number of words to be disambiguated, the mean polysemy, and the dimension of training, test and unlabeled corpora. Note that the organizers of the English task did not provide any unlabeled material. So for English we used a domain model built from a portion of BNC corpus, while for Spanish, Italian and Catalan we acquired DMs from the unlabeled corpora made available by the organizers.

	#w	*pol*	*# train*	*# test*	*# unlab*
Catalan	27	3.11	4469	2253	23935
English	57	6.47	7860	3944	-
Italian	45	6.30	5145	2439	74788
Spanish	46	3.30	8430	4195	61252

Table 2: Dataset descriptions

4.2 Kernel Combination

In this section we present an experiment to empirically study the kernel combination. The basic kernels (i.e. K_{BoW}, K_D, K_{Coll} and K_{PoS}) have been compared to the combined ones (i.e. K_{wsd} and K'_{wsd}) on the English lexical sample task.

The results are reported in Table 3. The results show that combining kernels significantly improves the performance of the system.

	K_D	K_{BoW}	K_{PoS}	K_{Coll}	K_{wsd}	K'_{wsd}
F1	65.5	63.7	62.9	66.7	**69.7**	**73.3**

Table 3: The performance (F1) of each basic kernel and their combination for English lexical sample task.

4.3 Portability and Performance

We evaluated the performance of K'_{wsd} and K_{wsd} on the lexical sample tasks described above. The results are showed in Table 4 and indicate that using DMs allowed K'_{wsd} to significantly outperform K_{wsd}.

In addition, K'_{wsd} turns out the best systems for all the tested Senseval-3 tasks.

Finally, the performance of K'_{wsd} are higher than the human agreement for the English and Spanish tasks[5].

Note that, in order to guarantee an uniform application to any language, we do not use any syntactic information provided by a parser.

4.4 Learning Curves

The Figures 1, 2, 3 and 4 show the learning curves evaluated on K'_{wsd} and K_{wsd} for all the lexical sample tasks.

The learning curves indicate that K'_{wsd} is far superior to K_{wsd} for all the tasks, even with few examples. The result is extremely promising, for it demonstrates that DMs allow to drastically reduce the amount of sense tagged data required for learning. It is worth noting, as reported in Table 5, that K'_{wsd} achieves the same performance of K_{wsd} using about half of the training data.

	% of training
English	54
Catalan	46
Italian	51
Spanish	50

Table 5: Percentage of sense tagged examples required by K'_{wsd} to achieve the same performance of K_{wsd} with full training.

5 Conclusion and Future Works

In this paper we presented a supervised algorithm for WSD, based on a combination of kernel functions. In particular we modeled domain and syntagmatic aspects of sense distinctions by defining respectively domain and syntagmatic kernels. The Domain kernel exploits Domain Models, acquired from "external" untagged corpora, to estimate the similarity among the contexts of the words to be disambiguated. The syntagmatic kernels evaluate the similarity between collocations.

We evaluated our algorithm on several Senseval-3 lexical sample tasks (i.e. English, Spanish, Italian and Catalan) significantly improving the state-ot-the-art for all of them. In addition, the performance

[5]It is not clear if the inter-annotator-agreement can be considerated the upper bound for a WSD system.

	MF	*Agreement*	*BEST*	K_{wsd}	K'_{wsd}	*DM+*
English	55.2	67.3	72.9	69.7	**73.3**	3.6
Catalan	66.3	93.1	85.2	85.2	**89.0**	3.8
Italian	18.0	89.0	53.1	53.1	**61.3**	8.2
Spanish	67.7	85.3	84.2	84.2	**88.2**	4.0

Table 4: Comparative evaluation on the lexical sample tasks. Columns report: the *Most Frequent* baseline, the *inter annotator agreement*, the *F1* of the best system at Senseval-3, the *F1* of K_{wsd}, the *F1* of K'_{wsd}, *DM+* (the improvement due to DM, i.e. $K'_{wsd} - K_{wsd}$).

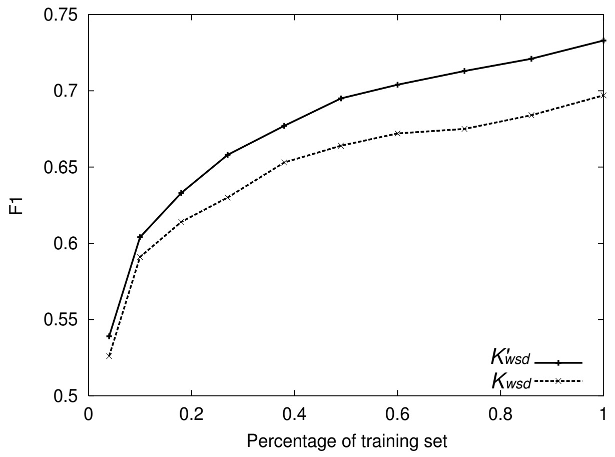


Figure 1: Learning curves for English lexical sample task.

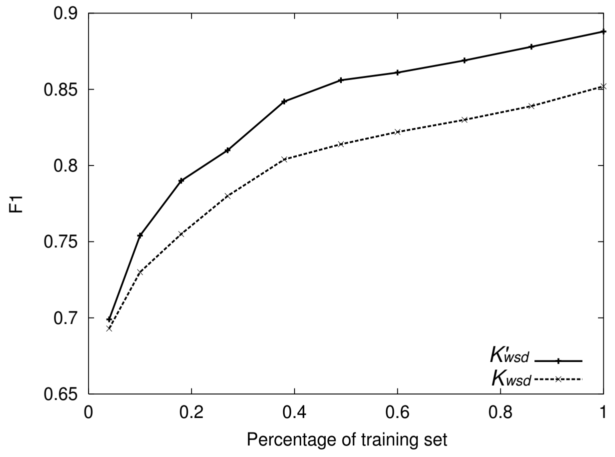


Figure 2: Learning curves for Catalan lexical sample task.

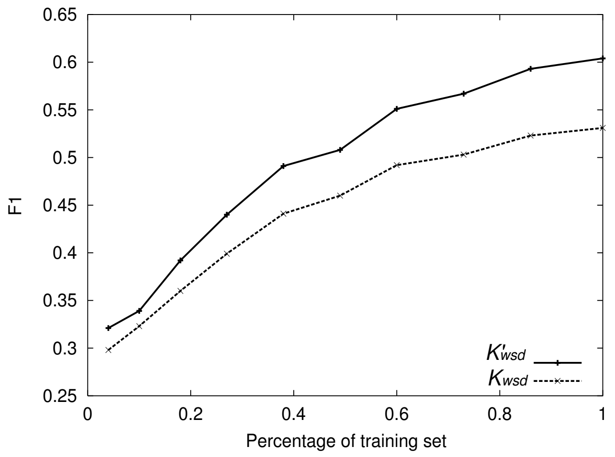


Figure 3: Learning curves for Italian lexical sample task.

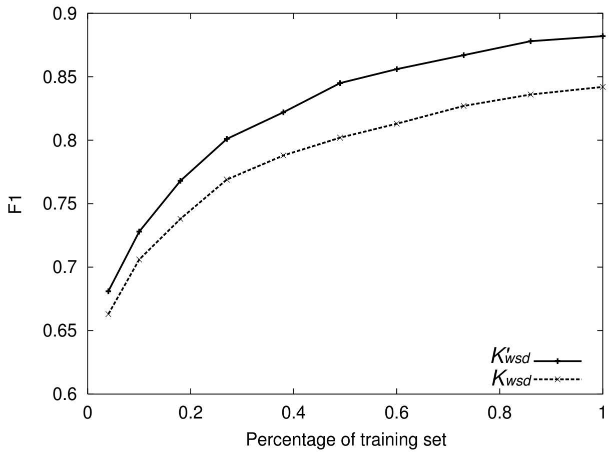


Figure 4: Learning curves for Spanish lexical sample task.

of our system outperforms the inter annotator agreement in both English and Spanish, achieving the upper bound performance.

We demonstrated that using external knowledge inside a supervised framework is a viable methodology to reduce the amount of training data required for learning. In our approach the external knowledge is represented by means of Domain Models automatically acquired from corpora in a totally unsupervised way. Experimental results show that the use of Domain Models allows us to reduce the amount of training data, opening an interesting research direction for all those NLP tasks for which the Knowledge Acquisition Bottleneck is a crucial problem. In particular we plan to apply the same methodology to Text Categorization, by exploiting the Domain Kernel to estimate the similarity among texts. In this implementation, our WSD system does not exploit syntactic information produced by a parser. For the future we plan to integrate such information by adding a tree kernel (i.e. a kernel function that evaluates the similarity among parse trees) to the kernel combination schema presented in this paper. Last but not least, we are going to apply our approach to develop supervised systems for all-words tasks, where the quantity of data available to train each word expert classifier is very low.

Acknowledgments

Alfio Gliozzo and Carlo Strapparava were partially supported by the EU project Meaning (IST-2001-34460). Claudio Giuliano was supported by the EU project Dot.Kom (IST-2001-34038). We would like to thank Oier Lopez de Lacalle for useful comments.

References

N. Cristianini and J. Shawe-Taylor. 2000. *An introduction to Support Vector Machines*. Cambridge University Press.

B. Decadt, V. Hoste, W. Daelemens, and A. van den Bosh. 2004. Gambl, genetic algorithm optimization of memory-based wsd. In *Proc. of Senseval-3*, Barcelona, July.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*.

A. Gliozzo, C. Strapparava, and I. Dagan. 2004. Unsupervised and supervised exploitation of semantic domains in lexical disambiguation. *Computer Speech and Language*, 18(3):275–299.

B. Magnini and G. Cavaglià. 2000. Integrating subject field codes into WordNet. In *Proceedings of LREC-2000*, pages 1413–1418, Athens, Greece, June.

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.

R. Mihalcea and P. Edmonds, editors. 2004. *Proceedings of SENSEVAL-3*, Barcelona, Spain, July.

R. Mihalcea and E. Faruque. 2004. Senselearner: Minimally supervised WSD for all words in open text. In *Proceedings of SENSEVAL-3*, Barcelona, Spain, July.

G. Salton and M.H. McGill. 1983. *Introduction to modern information retrieval*. McGraw-Hill, New York.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

S. Small. 1980. *Word Expert Parsing: A Theory of Distributed Word-based Natural Language Understanding*. Ph.D. Thesis, Department of Computer Science, University of Maryland.

C. Strapparava, A. Gliozzo, and C. Giuliano. 2004. Pattern abstraction and term similarity for word sense disambiguation: Irst at senseval-3. In *Proc. of SENSEVAL-3 Third International Workshop on Evaluation of Systems for the Semantic Analysis of Text*, pages 229–234, Barcelona, Spain, July.

S.K.M. Wong, W. Ziarko, and P.C.N. Wong. 1985. Generalized vector space model in information retrieval. In *Proceedings of the 8^{th} ACM SIGIR Conference*.

D. Yarowsky and R. Florian. 2002. Evaluating sense disambiguation across diverse parameter space. *Natural Language Engineering*, 8(4):293–310.

Improving Name Tagging by Reference Resolution and Relation Detection

Heng Ji **Ralph Grishman**
Department of Computer Science
New York University
New York, NY, 10003, USA
hengji@cs.nyu.edu grishman@cs.nyu.edu

Abstract

Information extraction systems incorporate multiple stages of linguistic analysis. Although errors are typically compounded from stage to stage, it is possible to reduce the errors in one stage by harnessing the results of the other stages. We demonstrate this by using the results of coreference analysis and relation extraction to reduce the errors produced by a Chinese name tagger. We use an N-best approach to generate multiple hypotheses and have them re-ranked by subsequent stages of processing. We obtained thereby a reduction of 24% in spurious and incorrect name tags, and a reduction of 14% in missed tags.

1 Introduction

Systems which extract relations or events from a document typically perform a number of types of linguistic analysis in preparation for information extraction. These include name identification and classification, parsing (or partial parsing), semantic classification of noun phrases, and coreference analysis. These tasks are reflected in the evaluation tasks introduced for MUC-6 (named entity, coreference, template element) and MUC-7 (template relation).

In most extraction systems, these stages of analysis are arranged *sequentially*, with each stage using the results of prior stages and generating a single analysis that gets enriched by each stage. This provides a simple modular organization for the extraction system.

Unfortunately, each stage also introduces a certain level of error into the analysis. Furthermore, these errors are compounded – for example, errors in name recognition may lead to errors in parsing. The net result is that the final output (relations or events) may be quite inaccurate.

This paper considers how interactions between the stages can be exploited to reduce the error rate. For example, the results of coreference analysis or relation identification may be helpful in name classification, and the results of relation or event extraction may be helpful in coreference.

Such interactions are not easily exploited in a simple sequential model … if name classification is performed at the beginning of the pipeline, it cannot make use of the results of subsequent stages. It may even be difficult to use this information *implicitly*, by using features which are also used in later stages, because the representation used in the initial stages is too limited.

To address these limitations, some recent systems have used more *parallel* designs, in which a single classifier (incorporating a wide range of features) encompasses what were previously several separate stages (Kambhatla, 2004; Zelenko et al., 2004). This can reduce the compounding of errors of the sequential design. However, it leads to a very large feature space and makes it difficult to select linguistically appropriate features for particular analysis tasks. Furthermore, because these decisions are being made in parallel, it becomes much harder to express interactions between the levels of analysis based on linguistic intuitions.

Proceedings of the 43rd Annual Meeting of the ACL, pages 411–418,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

In order to capture these interactions more explicitly, we have employed a sequential design in which multiple hypotheses are forwarded from each stage to the next, with hypotheses being reranked and pruned using the information from later stages. We shall apply this design to show how named entity classification can be improved by 'feedback' from coreference analysis and relation extraction. We shall show that this approach can capture these interactions in a natural and efficient manner, yielding a substantial improvement in name identification and classification.

2 Prior Work

A wide variety of trainable models have been applied to the name tagging task, including HMMs (Bikel et al., 1997), maximum entropy models (Borthwick, 1999), support vector machines (SVMs), and conditional random fields. People have spent considerable effort in engineering appropriate features to improve performance; most of these involve internal name structure or the immediate local context of the name.

Some other named entity systems have explored global information for name tagging. (Borthwick, 1999) made a second tagging pass which uses information on token sequences tagged in the first pass; (Chieu and Ng, 2002) used as features information about features assigned to other instances of the same token.

Recently, in (Ji and Grishman, 2004) we proposed a name tagging method which applied an SVM based on coreference information to filter the names with low confidence, and used coreference rules to correct and recover some names. One limitation of this method is that in the process of discarding many incorrect names, it also discarded some correct names. We attempted to recover some of these names by heuristic rules which are quite language specific. In addition, this single-hypothesis method placed an upper bound on recall.

Traditional statistical name tagging methods have generated a single name hypothesis. BBN proposed the N-Best algorithm for speech recognition in (Chow and Schwartz, 1989). Since then N-Best methods have been widely used by other researchers (Collins, 2002; Zhai et al., 2004).

In this paper, we tried to combine the advantages of the prior work, and incorporate broader knowledge into a more general re-ranking model.

3 Task and Terminology

Our experiments were conducted in the context of the ACE Information Extraction evaluations, and we will use the terminology of these evaluations:

entity: an object or a set of objects in one of the semantic categories of interest
mention: a reference to an entity (typically, a noun phrase)
name mention: a reference by name to an entity
nominal mention: a reference by a common noun or noun phrase to an entity
relation: one of a specified set of relationships between a pair of entities

The 2004 ACE evaluation had 7 types of entities, of which the most common were PER (persons), ORG (organizations), and GPE ('geo-political entities' – locations which are also political units, such as countries, counties, and cities). There were 7 types of relations, with 23 subtypes. Examples of these relations are "the CEO of Microsoft" (an *employ-exec* relation), "Fred's wife" (a *family* relation), and "a military base in Germany" (a *located* relation).

In this paper we look at the problem of identifying name mentions in Chinese text and classifying them as persons, organizations, or GPEs. Because Chinese has neither capitalization nor overt word boundaries, it poses particular problems for name identification.

4 Baseline System

4.1 Baseline Name Tagger

Our baseline name tagger consists of a HMM tagger augmented with a set of post-processing rules. The HMM tagger generally follows the Nymble model (Bikel et al, 1997), but with multiple hypotheses as output and a larger number of states (12) to handle name prefixes and suffixes, and transliterated foreign names separately. It operates on the output of a word segmenter from Tsinghua University.

Within each of the name class states, a statistical bigram model is employed, with the usual one-word-per-state emission. The various probabilities involve word co-occurrence, word features, and class probabilities. Then it uses A* search decoding to generate multiple hypotheses. Since these probabilities are estimated based on observations seen in a corpus, "back-off models" are used to reflect the strength of support for a given statistic, as for the Nymble system.

We also add post-processing rules to correct some omissions and systematic errors using name lists (for example, a list of all Chinese last names; lists of organization and location suffixes) and particular contextual patterns (for example, verbs occurring with people's names). They also deal with abbreviations and nested organization names.

The HMM tagger also computes the margin – the difference between the log probabilities of the top two hypotheses. This is used as a rough measure of confidence in the top hypothesis (see sections 5.3 and 6.2, below).

The name tagger used for these experiments identifies the three main ACE entity types: Person (PER), Organization (ORG), and GPE (names of the other ACE types are identified by a separate component of our system, not involved in the experiments reported here).

4.2 Nominal Mention Tagger

Our nominal mention tagger (noun group recognizer) is a maximum entropy tagger trained on the Chinese TreeBank from the University of Pennsylvania, supplemented by list matching.

4.3 Reference Resolver

Our baseline reference resolver goes through two successive stages: first, coreference rules will identify some high-confidence positive and negative mention pairs, in training data and test data; then the remaining samples will be used as input of a maximum entropy tagger. The features used in this tagger involve distance, string matching, lexical information, position, semantics, etc. We separate the task into different classifiers for different mention types (name / noun / pronoun). Then we incorporate the results from the relation tagger to adjust the probabilities from the classifiers. Finally we apply a clustering algorithm to combine them into entities (sets of coreferring mentions).

4.4 Relation Tagger

The relation tagger uses a k-nearest-neighbor algorithm. For both training and test, we consider all pairs of entity mentions where there is at most one other mention between the heads of the two mentions of interest[1]. Each training / test example consists of the pair of mentions and the sequence of intervening words. Associated with each training example is either one of the ACE relation types or no relation at all. We defined a distance metric between two examples based on

- whether the heads of the mentions match
- whether the ACE types of the heads of the mentions match (for example, both are people or both are organizations)
- whether the intervening words match

To tag a test example, we find the k nearest training examples (where k = 3) and use the distance to weight each neighbor, then select the most common class in the weighted neighbor set.

To provide a crude measure of the confidence of our relation tagger, we define two thresholds, D_{near} and D_{far}. If the average distance d to the nearest neighbors $d < D_{near}$, we consider this a *definite* relation. If $D_{near} < d < D_{far}$, we consider this a *possible* relation. If $d > D_{far}$, the tagger assumes that no relation exists (regardless of the class of the nearest neighbor).

5 Information from Coreference and Relations

Our system is processing a *document* consisting of multiple sentences. For each sentence, the name recognizer generates multiple hypotheses, each of which is an NE tagging of the entire sentence. The names in the hypothesis, plus the nouns in the categories of interest constitute the mention set for that hypothesis. Coreference resolution links these mentions, assigning each to an entity. In symbols:

S_i is the i-th sentence in the document.

H_i is the hypotheses set for S_i

h_{ij} is the j-th hypothesis in S_i

M_{ij} is the mention set for h_{ij}

m_{ijk} is the k-th mention in M_{ij}

e_{ijk} is the entity which m_{ijk} belongs to according to the current reference resolution results

5.1 Coreference Features

For each mention we compute seven quantities based on the results of name tagging and reference resolution:

[1] This constraint is relaxed for parallel structures such as "mention1, mention2, [and] mention3….”; in such cases there can be more than one intervening mention.

$CorefNum_{ijk}$ is the number of mentions in e_{ijk}

$WeightSum_{ijk}$ is the sum of all the link weights between m_{ijk} and other mentions in e_{ijk}, 0.8 for name-name coreference; 0.5 for apposition; 0.3 for other name-nominal coreference

$FirstMention_{ijk}$ is 1 if m_{ijk} is the first name mention in the entity; otherwise 0

$Head_{ijk}$ is 1 if m_{ijk} includes the head word of name; otherwise 0

$Withoutidiom_{ijk}$ is 1 if m_{ijk} is not part of an idiom; otherwise 0

$PERContext_{ijk}$ is the number of PER context words around a PER name such as a title or an action verb involving a PER

$ORGSuffix_{ijk}$ is 1 if ORG m_{ijk} includes a suffix word; otherwise 0

The first three capture evidence of the correctness of a name provided by reference resolution; for example, a name which is coreferenced with more other mentions is more likely to be correct. The last four capture local or name-internal evidence; for instance, that an organization name includes an explicit, organization-indicating suffix.

We then compute, for each of these seven quantities, the sum over all mentions k in a sentence, obtaining values for $CorefNum_{ij}$, $WeightSum_{ij}$, etc.:

$$CorefNum_{ij} = \sum_k CorefNum_{ijk} \quad \text{etc.}$$

Finally, we determine, for a given sentence and hypothesis, for each of these seven quantities, whether this quantity achieves the maximum of its values for this hypothesis:

$$BestCorefNum_{ij} \equiv CorefNum_{ij} = \max_q CorefNum_{iq} \quad \text{etc.}$$

We will use these properties of the hypothesis as features in assessing the quality of a hypothesis.

5.2 Relation Word Clusters

In addition to using relation information for reranking name hypotheses, we used the relation training corpus to build word clusters which could more directly improve name tagging. Name taggers rely heavily on words in the immediate context to identify and classify names; for example, specific job titles, occupations, or family relations can be used to identify people names. Such words are learned individually from the name tagger's training corpus. If we can provide the name tagger with clusters of related words, the tagger will be able to generalize from the examples in the training corpus to other words in the cluster.

The set of ACE relations includes several involving employment, social, and family relations. We gathered the words appearing as an argument of one of these relations in the training corpus, eliminated low-frequency terms and manually edited the ten resulting clusters to remove inappropriate terms. These were then combined with lists (of titles, organization name suffixes, location suffixes) used in the baseline tagger.

5.3 Relation Features

Because the performance of our relation tagger is not as good as our coreference resolver, we have used the results of relation detection in a relatively simple way to enhance name detection. The basic intuition is that a name which has been correctly identified is more likely to participate in a relation than one which has been erroneously identified.

For a given range of margins (from the HMM), the probability that a name in the first hypothesis is correct is shown in the following table, for names participating and not participating in a relation:

Margin	In Relation(%)	Not in Relation(%)
<4	90.7	55.3
<3	89.0	50.1
<2	86.9	42.2
<1.5	81.3	28.9
<1.2	78.8	23.1
<1	75.7	19.0
<0.5	66.5	14.3

Table 1 Probability of a name being correct

Table 1 confirms that names participating in relations are much more likely to be correct than names that do not participate in relations. We also see, not surprisingly, that these probabilities are strongly affected by the HMM hypothesis margin (the difference in log probabilities) between the first hypothesis and the second hypothesis. So it is natural to use participation in a relation (coupled with a margin value) as a valuable feature for reranking name hypotheses.

Let m_{ijk} be the k-th name mention for hypothesis h_{ij} of sentence; then we define:

$$Inrelation_{ijk} \begin{array}{l} = 1 \text{ if } m_{ijk} \text{ is in a definite relation} \\ = 0 \text{ if } m_{ijk} \text{ is in a possible relation} \\ = -1 \text{ if } m_{ijk} \text{ is not in a relation} \end{array}$$

$$Inrelation_{ij} = \sum_k Inrelation_{ijk}$$

$$Mostrelated_{ij} \equiv (Inrelation_{ij} = \max_q Inrelation_{iq})$$

Finally, to capture the interaction with the margin, we let z_i = the margin for sentence S_i and divide the range of values of z_i into six intervals Mar_1, … Mar_6. And we define the hypothesis ranking information: $FirstHypothesis_{ij}$ = 1 if j =1; otherwise 0.

We will use as features for ranking h_{ij} the conjunction of $Mostrelated_{ij}$, $z_i \in Mar_p$ (p = 1, …, 6), and $FirstHypothesis_{ij}$.

6 Using the Information from Coreference and Relations

6.1 Word Clustering based on Relations

As we described in section 5.2, we can generate word clusters based on relation information. If a word is not part of a relation cluster, we consider it an independent (1-word) cluster.

The Nymble name tagger (Bikel et al., 1999) relies on a multi-level linear interpolation model for backoff. We extended this model by adding a level from word to cluster, so as to estimate more reliable probabilities for words in these clusters. Table 2 shows the extended backoff model for each of the three probabilities used by Nymble.

Transition Probability	First-Word Emission Probability	Non-First-Word Emission Probability
$P(NC_2 \mid NC_1, <w_1, f_1>)$	$P(<w_2,f_2> \mid NC_1, NC_2)$	$P(<w_2,f_2> \mid <w_1,f_1>, NC_2)$
	$P(<Cluster_2,f_2> \mid NC_1, NC_2)$	$P(<Cluster_2,f_2> \mid <w_1,f_1>, NC_2)$
$P(NC_2 \mid NC_1, <Cluster_1, f_1>)$	$P(<Cluster_2,f_2> \mid <+begin+, other>, NC_2)$	$P(<Cluster_2,f_2> \mid <Cluster_1,f_1>, NC_2)$
$P(NC_2 \mid NC_1)$	$P(<Cluster_2, f_2> \mid NC_2)$	
$P(NC_2)$	$P(Cluster_2 \mid NC_2) * P(f_2 \mid NC_2)$	
1/#(name classes)	1/#(cluster) * 1/#(word features)	

Table2 Extended Backoff Model

6.2 Pre-pruning by Margin

The HMM tagger produces the N best hypotheses for each sentence.[2] In order to decide when we need to rely on global (coreference and relation) information for name tagging, we want to have some assessment of the confidence that the name tagger has in the first hypothesis. In this paper, we use the margin for this purpose. A large margin indicates greater confidence that the first hypothesis is correct.[3] So if the margin of a sentence is above a threshold, we select the first hypothesis, dropping the others and by-passing the reranking.

6.3 Re-ranking based on Coreference

We described in section 5.1, above, the coreference features which will be used for reranking the hypotheses after pre-pruning. A maximum entropy model for re-ranking these hypotheses is then trained and applied as follows:

Training

```
1. Use K-fold cross-validation to generate multiple name tagging hypotheses for each document in the training data $D_{train}$ (in each of the K iterations, we use K-1 subsets to train the HMM and then generate hypotheses from the Kth subset).
2. For each document d in $D_{train}$, where d includes n sentences $S_1…S_n$
      For i = 1…n, let m = the number of hypotheses for $S_i$
      (1) Pre-prune the candidate hypotheses using the HMM margin
      (2) For each hypothesis $h_{ij}$, j = 1…m
            (a) Compare $h_{ij}$ with the key, set the prediction $Value_{ij}$ "Best" or "Not Best"
            (b) Run the Coreference Resolver on $h_{ij}$ and the best hypothesis for each of the other sentences, generate entity results for each candidate name in $h_{ij}$
            (c) Generate a coreference feature vector $V_{ij}$ for $h_{ij}$
            (d) Output $V_{ij}$ and $Value_{ij}$
```

[2] We set different N = 5, 10, 20 or 30 for different margin ranges, by cross-validation checking the training data about the ranking position of the best hypothesis for each sentence. With this N, optimal reranking (selecting the best hypothesis among the N best) would yield Precision = 96.9 Recall = 94.5 F = 95.7 on our test corpus.

[3] Similar methods based on HMM margins were used by (Scheffer et al., 2001).

3. Train Maxent Re-ranking system on all V_{ij} and $Value_{ij}$

Test

1. Run the baseline name tagger to generate multiple name tagging hypotheses for each document in the test data D_{test}
2. For each document d in D_{test}, where d includes n sentences $S_1 \ldots S_n$
   (1) Initialize: Dynamic input of coreference resolver $H = \{h_{i\text{-best}} \mid i = 1 \ldots n$, $h_{i\text{-best}}$ is the current best hypothesis for $S_i\}$
   (2) For i = 1…n, assume m = the number of hypotheses for S_i
      (a) Pre-prune the candidate hypotheses using the HMM margin
      (b) For each hypothesis h_{ij}, j = 1…m
         - $h_{i\text{-best}} = h_{ij}$
         - Run the Coreference Resolver on H, generate entity results for each name candidate in h_{ij}
         - Generate a coreference feature vector V_{ij} for h_{ij}
         - Run Maxent Re-ranking system on V_{ij}, produce $Prob_{ij}$ of "Best" value
      (c) $h_{i\text{-best}}$ = the hypothesis with highest $Prob_{ij}$ of "Best" value, update H and output $h_{i\text{-best}}$

6.4 Re-ranking based on Relations

From the above first-stage re-ranking by coreference, for each hypothesis we got the probability of its being the best one. By using these results and relation information we proceed to a second-stage re-ranking. As we described in section 5.3, the information of "in relation or not" can be used together with margin as another important measure of confidence.

In addition, we apply the mechanism of weighted voting among hypotheses (Zhai et al., 2004) as an additional feature in this second-stage re-ranking. This approach allows all hypotheses to vote on a possible name output. A recognized name is considered correct only when it occurs in more than 30 percent of the hypotheses (weighted by their probability).

In our experiments we use the probability produced by the HMM, $prob_{ij}$, for hypothesis h_{ij}. We normalize this probability weight as:

$$W_{ij} = \frac{\exp(prob_{ij})}{\sum_q \exp(prob_{iq})}$$

For each name mention m_{ijk} in h_{ij}, we define:

$$Occur_q(m_{ijk}) = 1 \text{ if } m_{ijk} \text{ occurs in } h_q$$
$$= 0 \text{ otherwise}$$

Then we count its voting value as follows:

$$Voting_{ijk} \text{ is 1 if } \sum_q W_{iq} \times Occur_q(m_{ijk}) > 0.3;$$
$$\text{otherwise 0.}$$

The voting value of h_{ij} is:

$$Voting_{ij} = \sum_k Voting_{ijk}$$

Finally we define the following voting feature:

$$BestVoting_{ij} \equiv (Voting_{ij} = \max_q Voting_{iq})$$

This feature is used, together with the features described at the end of section 5.3 and the probability score from the first stage, for the second-stage maxent re-ranking model.

One appeal of the above two re-ranking algorithms is its flexibility in incorporating features into a learning model: essentially any coreference or relation features which might be useful in discriminating good from bad structures can be included.

7 System Pipeline

Combining all the methods presented above, the flow of our final system is shown in figure 1.

8 Evaluation Results

8.1 Training and Test Data

We took 346 documents from the 2004 ACE training corpus and official test set, including both broadcast news and newswire, as our blind test set. To train our name tagger, we used the Beijing University Insititute of Computational Linguistics corpus – 2978 documents from the People's Daily in 1998 – and 667 texts in the training corpus for the 2003 & 2004 ACE evaluation. Our reference resolver is trained on these 667 ACE texts. The relation tagger is trained on 546 ACE 2004 texts, from which we also extracted the relation clusters. The test set included 11715 names: 3551 persons, 5100 GPEs and 3064 organizations.

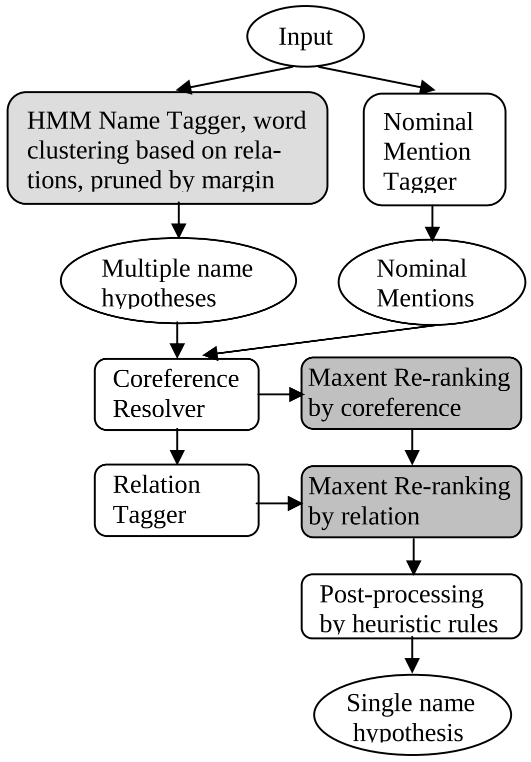

Input
HMM Name Tagger, word clustering based on relations, pruned by margin
Nominal Mention Tagger
Multiple name hypotheses
Nominal Mentions
Coreference Resolver
Maxent Re-ranking by coreference
Relation Tagger
Maxent Re-ranking by relation
Post-processing by heuristic rules
Single name hypothesis

Figure 1 System Flow

8.2 Overall Performance Comparison

Table 3 shows the performance of the baseline system; Table 4 is the system with relation word clusters; Table 5 is the system with both relation clusters and re-ranking based on coreference features; and Table 6 is the whole system with second-stage re-ranking using relations.

The results indicate that relation word clusters help to improve the precision and recall of most name types. Although the overall gain in F-score is small (0.7%), we believe further gain can be achieved if the relation corpus is enlarged in the future. The re-ranking using the coreference features had the largest impact, improving precision and recall consistently for all types. Compared to our system in (Ji and Grishman, 2004), it helps to distinguish the good and bad hypotheses without any loss of recall. The second-stage re-ranking using the relation participation feature yielded a small further gain in F score for each type, improving precision at a slight cost in recall.

The overall system achieves a 24.1% relative reduction on the spurious and incorrect tags, and 14.3% reduction in the missing rate over a state-of-the-art baseline HMM trained on the same material. Furthermore, it helps to disambiguate many name type errors: the number of cases of type confusion in name classification was reduced from 191 to 102.

Name	Precision	Recall	F
PER	88.6	89.2	88.9
GPE	88.1	84.9	86.5
ORG	88.8	87.3	88.0
ALL	88.4	86.7	87.5

Table 3 Baseline Name Tagger

Name	Precision	Recall	F
PER	89.4	90.1	89.7
GPE	88.9	85.8	89.4
ORG	88.7	87.4	88.0
ALL	89.0	87.4	88.2

Table 4 Baseline + Word Clustering by Relation

Name	Precision	Recall	F
PER	90.1	91.2	90.5
GPE	89.7	86.8	88.2
ORG	90.6	89.8	90.2
ALL	90.0	88.8	89.4

Table 5 Baseline + Word Clustering by Relation + Re-ranking by Coreference

Name	Precision	Recall	F
PER	90.7	91.0	90.8
GPE	91.2	86.9	89.0
ORG	91.7	89.1	90.4
ALL	91.2	88.6	89.9

Table 6 Baseline + Word Clustering by Relation + Re-ranking by Coreference + Re-ranking by Relation

In order to check how robust these methods are, we conducted significance testing (sign test) on the 346 documents. We split them into 5 folders, 70 documents in each of the first four folders and 66 in the fifth folder. We found that each enhancement (word clusters, coreference reranking, relation reranking) produced an improvement in F score for each folder, allowing us to reject the hypothesis that these improvements were random at a 95% confidence level. The overall F-measure improvements (using all enhancements) for the 5 folders were: 2.3%, 1.6%, 2.1%, 3.5%, and 2.1%.

9 Conclusion

This paper explored methods for exploiting the interaction of analysis components in an information extraction system to reduce the error rate of individual components. The ACE task hierarchy provided a good opportunity to explore these interactions, including the one presented here between reference resolution/relation detection and name tagging. We demonstrated its effectiveness for Chinese name tagging, obtaining an absolute improvement of 2.4% in F-measure (a reduction of 19% in the (1 – F) error rate). These methods are quite low-cost because we don't need any extra resources or components compared to the baseline information extraction system.

Because no language-specific rules are involved and no additional training resources are required, we expect that the approach described here can be straightforwardly applied to other languages. It should also be possible to extend this re-ranking framework to other levels of analysis in information extraction –- for example, to use event detection to improve name tagging; to incorporate subtype tagging results to improve name tagging; and to combine name tagging, reference resolution and relation detection to improve nominal mention tagging. For Chinese (and other languages without overt word segmentation) it could also be extended to do character-based name tagging, keeping multiple segmentations among the N-Best hypotheses. Also, as information extraction is extended to capture cross-document information, we should expect further improvements in performance of the earlier stages of analysis, including in particular name identification.

For some levels of analysis, such as name tagging, it will be natural to apply lattice techniques to organize the multiple hypotheses, at some gain in efficiency.

Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency under Grant N66001-04-1-8920 from SPAWAR San Diego, and by the National Science Foundation under Grant 03-25657. This paper does not necessarily reflect the position or the policy of the U.S. Government.

References

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. *Nymble: a high-performance Learning Name-finder*. Proc. Fifth Conf. on Applied Natural Language Processing, Washington, D.C.

Andrew Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. Dissertation, Dept. of Computer Science, New York University.

Hai Leong Chieu and Hwee Tou Ng. 2002. *Named Entity Recognition: A Maximum Entropy Approach Using Global Information.* Proc.: 17th Int'l Conf. on Computational Linguistics (COLING 2002), Taipei, Taiwan.

Yen-Lu Chow and Richard Schwartz. 1989. *The N-Best Algorithm: An efficient Procedure for Finding Top N Sentence Hypotheses.* Proc. DARPA Speech and Natural Language Workshop

Michael Collins. 2002. *Ranking Algorithms for Named-Entity Extraction: Boosting and the Voted Perceptron*. Proc. ACL 2002

Heng Ji and Ralph Grishman. 2004. *Applying Coreference to Improve Name Recognition.* Proc. ACL 2004 Workshop on Reference Resolution and Its Applications, Barcelona, Spain

N. Kambhatla. 2004. *Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations*. Proc. ACL 2004.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. *Active Hidden Markov Models for Information Extraction*. Proc. Int'l Symposium on Intelligent Data Analysis (IDA-2001).

Dmitry Zelenko, Chinatsu Aone, and Jason Tibbets. 2004. *Binary Integer Programming for Information Extraction*. ACE Evaluation Meeting, September 2004, Alexandria, VA.

Lufeng Zhai, Pascale Fung, Richard Schwartz, Marine Carpuat, and Dekai Wu. 2004. *Using N-best Lists for Named Entity Recognition from Chinese Speech*. Proc. NAACL 2004 (Short Papers)

Extracting Relations with Integrated Information Using Kernel Methods

Shubin Zhao **Ralph Grishman**
Department of Computer Science
New York University
715 Broadway, 7th Floor, New York, NY 10003
shubinz@cs.nyu.edu grishman@cs.nyu.edu

Abstract

Entity relation detection is a form of information extraction that finds predefined relations between pairs of entities in text. This paper describes a relation detection approach that combines clues from different levels of syntactic processing using kernel methods. Information from three different levels of processing is considered: tokenization, sentence parsing and deep dependency analysis. Each source of information is represented by kernel functions. Then composite kernels are developed to integrate and extend individual kernels so that processing errors occurring at one level can be overcome by information from other levels. We present an evaluation of these methods on the 2004 ACE relation detection task, using Support Vector Machines, and show that each level of syntactic processing contributes useful information for this task. When evaluated on the official test data, our approach produced very competitive ACE value scores. We also compare the SVM with KNN on different kernels.

1 Introduction

Information extraction subsumes a broad range of tasks, including the extraction of entities, relations and events from various text sources, such as newswire documents and broadcast transcripts. One such task, *relation detection*, finds instances of predefined relations between pairs of *entities*, such as a *Located-In* relation between the entities *Centre College* and *Danville, KY* in the phrase *Centre College in Danville, KY*. The 'entities' are the individuals of selected semantic types (such as people, organizations, countries, …) which are referred to in the text.

Prior approaches to this task (Miller et al., 2000; Zelenko et al., 2003) have relied on partial or full syntactic analysis. Syntactic analysis can find relations not readily identified based on sequences of tokens alone. Even 'deeper' representations, such as logical syntactic relations or predicate-argument structure, can in principle capture additional generalizations and thus lead to the identification of additional instances of relations. However, a general problem in Natural Language Processing is that as the processing gets deeper, it becomes less accurate. For instance, the current accuracy of tokenization, chunking and sentence parsing for English is about 99%, 92%, and 90% respectively. Algorithms based solely on deeper representations inevitably suffer from the errors in computing these representations. On the other hand, low level processing such as tokenization will be more accurate, and may also contain useful information missed by deep processing of text. Systems based on a single level of representation are forced to choose between shallower representations, which will have fewer errors, and deeper representations, which may be more general.

Based on these observations, Zhao et al. (2004) proposed a discriminative model to combine information from different syntactic sources using a kernel SVM (Support Vector Machine). We showed that adding sentence level word trigrams as global information to local dependency context boosted the performance of finding slot fillers for management succession events. This paper describes an extension of this approach to the identification of entity relations, in which syntactic information from sentence tokenization, parsing and deep dependency analysis is combined using kernel methods. At each level, kernel functions (or kernels) are developed to represent the syntactic information. Five kernels have been developed for this task, including two at the surface level, one at the parsing level and two at the deep dependency level. Our experiments show that each level of processing may contribute useful clues for this task, including surface information like word bigrams. Adding kernels one by one continuously improves performance. The experiments were carried out on the ACE RDR (Relation Detection and Recognition) task with annotated entities. Using SVM as a classifier along with the full composite kernel produced the best performance on this task. This paper will also show a comparison of SVM and KNN (k-Nearest-Neighbors) under different kernel setups.

2 Kernel Methods

Many machine learning algorithms involve only the dot product of vectors in a feature space, in which each vector represents an object in the object domain. Kernel methods (Muller et al., 2001) can be seen as a generalization of feature-based algorithms, in which the dot product is replaced by a kernel function (or kernel) $\Psi(X,Y)$ between two vectors, or even between two objects. Mathematically, as long as $\Psi(X,Y)$ is symmetric and the kernel matrix formed by Ψ is positive semi-definite, it forms a valid dot product in an implicit Hilbert space. In this implicit space, a kernel can be broken down into features, although the dimension of the feature space could be infinite.

Normal feature-based learning can be implemented in kernel functions, but we can do more than that with kernels. First, there are many well-known kernels, such as polynomial and radial basis kernels, which extend normal features into a high order space with very little computational cost. This could make a linearly non-separable problem separable in the high order feature space. Second, kernel functions have many nice combination properties: for example, the sum or product of existing kernels is a valid kernel. This forms the basis for the approach described in this paper. With these combination properties, we can combine individual kernels representing information from different sources in a principled way.

Many classifiers can be used with kernels. The most popular ones are SVM, KNN, and voted perceptrons. Support Vector Machines (Vapnik, 1998; Cristianini and Shawe-Taylor, 2000) are linear classifiers that produce a separating hyperplane with largest margin. This property gives it good generalization ability in high-dimensional spaces, making it a good classifier for our approach where using all the levels of linguistic clues could result in a huge number of features. Given all the levels of features incorporated in kernels and training data with target examples labeled, an SVM can pick up the features that best separate the targets from other examples, no matter which level these features are from. In cases where an error occurs in one processing result (especially deep processing) and the features related to it become noisy, SVM may pick up clues from other sources which are not so noisy. This forms the basic idea of our approach. Therefore under this scheme we can overcome errors introduced by one processing level; more particularly, we expect accurate low level information to help with less accurate deep level information.

3 Related Work

Collins et al. (1997) and Miller et al. (2000) used statistical parsing models to extract relational facts from text, which avoided pipeline processing of data. However, their results are essentially based on the output of sentence parsing, which is a deep processing of text. So their approaches are vulnerable to errors in parsing. Collins et al. (1997) addressed a simplified task within a confined context in a target sentence.

Zelenko et al. (2003) described a recursive kernel based on shallow parse trees to detect *person-affiliation* and *organization-location* relations, in which a relation example is the least common subtree containing two entity nodes. The kernel matches nodes starting from the roots of two subtrees and going recursively to the leaves. For each pair of nodes, a subsequence kernel on their child nodes is invoked, which matches either contiguous or non-contiguous subsequences of node. Compared with full parsing, shallow parsing is more reliable. But this model is based solely on the output of shallow parsing so it is still vulnerable to irrecoverable parsing errors. In their experiments, incorrectly parsed sentences were eliminated.

Culotta and Sorensen (2004) described a slightly generalized version of this kernel based on dependency trees. Since their kernel is a recursive match from the root of a dependency tree down to the leaves where the entity nodes reside, a successful match of two relation examples requires their entity nodes to be at the same depth of the tree. This is a strong constraint on the matching of syntax so it is not surprising that the model has good precision but very low recall. In their solution a bag-of-words kernel was used to compensate for this problem. In our approach, more flexible kernels are used to capture regularization in syntax, and more levels of syntactic information are considered.

Kambhatla (2004) described a Maximum Entropy model using features from various syntactic sources, but the number of features they used is limited and the selection of features has to be a manual process.[1] In our model, we use kernels to incorporate more syntactic information and let a Support Vector Machine decide which clue is crucial. Some of the kernels are extended to generate high order features. We think a discriminative classifier trained with all the available syntactic features should do better on the sparse data.

4 Kernel Relation Detection

4.1 ACE Relation Detection Task

ACE (Automatic Content Extraction)[2] is a research and development program in information extraction sponsored by the U.S. Government. The 2004 evaluation defined seven major types of relations between seven types of entities. The entity types are PER (Person), ORG (Organization), FAC (Facility), GPE (Geo-Political Entity: countries, cities, etc.), LOC (Location), WEA (Weapon) and VEH (Vehicle). Each mention of an entity has a mention type: NAM (proper name), NOM (nominal) or PRO (pronoun); for example *George W. Bush*, *the president* and *he* respectively. The seven relation types are EMP-ORG (Employment/Membership/Subsidiary), PHYS (Physical), PER-SOC (Personal/Social), GPE-AFF (GPE-Affiliation), Other-AFF (Person/ORG Affiliation), ART (Agent-Artifact) and DISC (Discourse). There are also 27 relation subtypes defined by ACE, but this paper only focuses on detection of relation types. Table 1 lists examples of each relation type.

Type	Example
EMP-ORG	*the CEO of Microsoft*
PHYS	*a military base in Germany*
GPE-AFF	*U.S. businessman*
PER-SOC	*a spokesman for the senator*
DISC	*many of these people*
ART	*the makers of the Kursk*
Other-AFF	*Cuban-American people*

Table 1. ACE relation types and examples. The heads of the two entity arguments in a relation are marked. Types are listed in decreasing order of frequency of occurrence in the ACE corpus.

Figure 1 shows a sample newswire sentence, in which three relations are marked. In this sentence, we expect to find a PHYS relation between *Hezbollah forces* and *areas,* a PHYS relation between *Syrian troops* and *areas* and an EMP-ORG relation between *Syrian troops* and *Syrian.* In our approach, input text is preprocessed by the Charniak sentence parser (including tokenization and POS tagging) and the GLARF (Meyers et al., 2001) dependency analyzer produced by NYU. Based on treebank parsing, GLARF produces labeled deep dependencies between words (syntactic relations such as logical subject and logical object). It handles linguistic phenomena like passives, relatives, reduced relatives, conjunctions, etc.

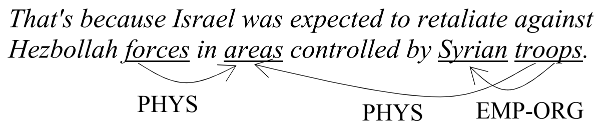

That's because Israel was expected to retaliate against
Hezbollah forces in areas controlled by Syrian troops.
PHYS
PHYS
EMP-ORG

Figure 1. Example sentence from newswire text

4.2 Definitions

In our model, kernels incorporate information from

[1] Kambhatla also evaluated his system on the ACE relation detection task, but the results are reported for the 2003 task, which used different relations and different training and test data, and did not use hand-annotated entities, so they cannot be readily compared to our results.

[2] Task description: http://www.itl.nist.gov/iad/894.01/tests/ace/
ACE guidelines: http://www.ldc.upenn.edu/Projects/ACE/

tokenization, parsing and deep dependency analysis. A relation candidate R is defined as

$R = (arg_1, arg_2, seq, link, path)$,

where arg_1 and arg_2 are the two entity arguments which may be related; $seq=(t_1, t_2, \ldots, t_n)$ is a token vector that covers the arguments and intervening words; $link=(t_1, t_2, \ldots, t_m)$ is also a token vector, generated from *seq* and the parse tree; *path* is a dependency path connecting arg_1 and arg_2 in the dependency graph produced by GLARF. *path* can be empty if no such dependency path exists. The difference between *link* and *seq* is that *link* only retains the "important" words in *seq* in terms of syntax. For example, all noun phrases occurring in *seq* are replaced by their heads. Words and constituent types in a stop list, such as time expressions, are also removed.

A token T is defined as a string triple,

$T = (word, pos, base)$,

where *word*, *pos* and *base* are strings representing the word, part-of-speech and morphological base form of T. Entity is a token augmented with other attributes,

$E = (tk, type, subtype, mtype)$,

where *tk* is the token associated with E; *type*, *subtype* and *mtype* are strings representing the entity type, subtype and mention type of E. The subtype contains more specific information about an entity. For example, for a GPE entity, the subtype tells whether it is a country name, city name and so on. Mention type includes NAM, NOM and PRO.

It is worth pointing out that we always treat an entity as a single token: for a nominal, it refers to its head, such as *boys* in *the two boys*; for a proper name, all the words are connected into one token, such as *Bashar_Assad*. So in a relation example R whose *seq* is $(t_1, t_2, \ldots, t_n)$, it is always true that $arg_1=t_1$ and $arg_2=t_n$. For names, the base form of an entity is its ACE type (person, organization, etc.). To introduce dependencies, we define a dependency token to be a token augmented with a vector of dependency arcs,

$DT=(word, pos, base, dseq)$,

where $dseq = (arc_1, \ldots, arc_n)$. A dependency arc is

$ARC = (w, dw, label, e)$,

where w is the current token; dw is a token connected by a dependency to w; and *label* and e are the role label and direction of this dependency arc respectively. From now on we upgrade the type of *tk* in arg_1 and arg_2 to be dependency tokens. Finally, *path* is a vector of dependency arcs,

$path = (arc_1, \ldots, arc_l)$,

where l is the length of the path and arc_i ($1 \leq i \leq l$) satisfies $arc_1.w=arg_1.tk$, $arc_{i+1}.w=arc_i.dw$ and $arc_l.dw=arg_2.tk$. So *path* is a chain of dependencies connecting the two arguments in R. The arcs in it do not have to be in the same direction.

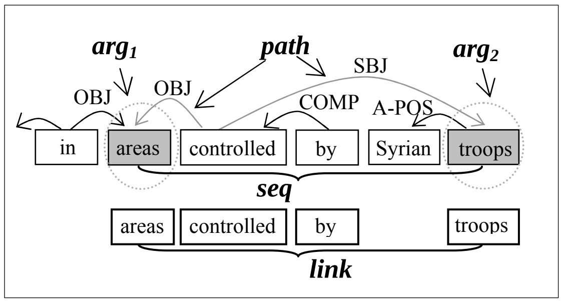


Figure 2. Illustration of a relation example R. The *link* sequence is generated from *seq* by removing some unimportant words based on syntax. The dependency links are generated by GLARF.

Figure 2 shows a relation example generated from the text "… *in areas controlled by Syrian troops*". In this relation example R, $\boldsymbol{arg_1}$ is *(("areas", "NNS", "area",* ***dseq****), "LOC", "Region", "NOM")*, and $\boldsymbol{arg_1.dseq}$ is *((OBJ, areas, in, 1), (OBJ, areas, controlled, 1))*. $\boldsymbol{arg_2}$ is *(("troops", "NNS", "troop",* ***dseq****), "ORG", "Government", "NOM")* and $\boldsymbol{arg_2.dseq}$ = *((A-POS, troops, Syrian, 0), (SBJ, troops, controlled, 1))*. ***path*** is *((OBJ, areas, controlled, 1), (SBJ, controlled, troops, 0))*. The value 0 in a dependency arc indicates forward direction from w to dw, and 1 indicates backward direction. The *seq* and *link* sequences of R are shown in Figure 2.

Some relations occur only between very restricted types of entities, but this is not true for every type of relation. For example, PER-SOC is a relation mainly between two person entities, while PHYS can happen between any type of entity and a GPE or LOC entity.

4.3 Syntactic Kernels

In this section we will describe the kernels designed for different syntactic sources and explain the intuition behind them.

We define two kernels to match relation examples at surface level. Using the notation just defined, we can write the two surface kernels as follows:

1) Argument kernel

$$\psi_1(R_1,R_2) = \sum_{i=1,2} K_E(R_1.\arg_i, R_2.\arg_i),$$

where K_E is a kernel that matches two entities,

$$K_E(E_1,E_2) = K_T(E_1.tk, E_2.tk) + I(E_1.type, E_2.type) + I(E_1.subtype, E_2.subtype) + I(E_1.mtype, E_2.mtype)$$

$$K_T(T_1,T_2) = I(T_1.word, T_2.word) + I(T_1.pos, T_2.pos) + I(T_1.base, T_2.base)$$

K_T is a kernel that matches two tokens. *I(x, y)* is a binary string match operator that gives 1 if x=*y* and 0 otherwise. Kernel Ψ_1 matches attributes of two entity arguments respectively, such as type, subtype and lexical head of an entity. This is based on the observation that there are type constraints on the two arguments. For instance PER-SOC is a relation mostly between two person entities. So the attributes of the entities are crucial clues. Lexical information is also important to distinguish relation types. For instance, in the phrase *U.S. president* there is an EMP-ORG relation between *president* and *U.S.*, while in *a U.S. businessman* there is a GPE-AFF relation between *businessman* and *U.S.*

2) Bigram kernel

$$\psi_2(R_1,R_2) = K_{seq}(R_1.seq, R_2.seq),$$

where

$$K_{seq}(seq, seq') = \sum_{0\le i<seq.len} \sum_{0\le j<seq'.len} (K_T(tk_i, tk'_j) + K_T(<tk_i, tk_{i+1}>, <tk'_j, tk'_{j+1}>))$$

Operator $<t_1, t_2>$ concatenates all the string elements in tokens t_1 and t_2 to produce a new token. So Ψ_2 is a kernel that simply matches unigrams and bigrams between the *seq* sequences of two relation examples. The information this kernel provides is faithful to the text.

3) Link sequence kernel

$$\psi_3(R_1,R_2) = K_{link}(R_1.link, R_2.link) = \sum_{0\le i<\min\_len} K_T(R_1.link.tk_i, R_2.link.tk_i),$$

where *min_len* is the length of the shorter link sequence in R_1 and R_2. Ψ_3 is a kernel that matches token by token between the link sequences of two relation examples. Since relations often occur in a short context, we expect many of them have similar link sequences.

4) Dependency path kernel

$$\psi_4(R_1,R_2) = K_{path}(R_1.path, R_2.path),$$

where

$$K_{path}(path, path') = \sum_{0\le i<path.len} \sum_{0\le j<path'.len} ((I(arc_i.label, arc'_j.label) + K_T(arc_i.dw, arc'_j.dw)) \times I(arc_i.e, arc'_j.e)$$

Intuitively the dependency path connecting two arguments could provide a high level of syntactic regularization. However, a complete match of two dependency paths is rare. So this kernel matches the component arcs in two dependency paths in a pairwise fashion. Two arcs can match only when they are in the same direction. In cases where two paths do not match exactly, this kernel can still tell us how similar they are. In our experiments we placed an upper bound on the length of dependency paths for which we computed a non-zero kernel.

5) Local dependency

$$\psi_5(R_1,R_2) = \sum_{i=1,2} K_D(R_1.\arg_i.dseq, R_2.\arg_i.dseq),$$

where

$$K_D(dseq, dseq') = \sum_{0\le i<dseq.len} \sum_{0\le j<dseq'.len} (I(arc_i.label, arc'_j.label) + K_T(arc_i.dw, arc'_j.dw)) \times I(arc_i.e, arc'_j.e)$$

This kernel matches the local dependency context around the relation arguments. This can be helpful especially when the dependency path between arguments does not exist. We also hope the dependencies on each argument may provide some useful clues about the entity or connection of the entity to the context outside of the relation example.

4.4 Composite Kernels

Having defined all the kernels representing shallow and deep processing results, we can define composite kernels to combine and extend the individual kernels.

1) Polynomial extension

$$\Phi_1(R_1,R_2) = (\psi_1 + \psi_3) + (\psi_1 + \psi_3)^2 / 4$$

This kernel combines the argument kernel Ψ_1 and link kernel Ψ_3 and applies a second-degree polynomial kernel to extend them. The combination of Ψ_1 and Ψ_3 covers the most important clues for this task: information about the two arguments and the word link between them. The polynomial extension is equivalent to adding pairs of features as new features. Intuitively this introduces new features like: the subtype of the first argument is a country name and the word of the second argument is *president*, which could be a good clue for an EMP-ORG relation. The polynomial kernel is down weighted by a normalization factor because we do not want the high order features to overwhelm the original ones. In our experiment, using polynomial kernels with degree higher than 2 does not produce better results.

2) Full kernel

$$\Phi_2(R_1, R_2) = \Phi_1 + \alpha\psi_4 + \beta\psi_5 + \chi\psi_2$$

This is the final kernel we used for this task, which is a combination of all the previous kernels. In our experiments, we set all the scalar factors to 1. Different values were tried, but keeping the original weight for each kernel yielded the best results for this task.

All the individual kernels we designed are explicit. Each kernel can be seen as a matching of features and these features are enumerable on the given data. So it is clear that they are all valid kernels. Since the kernel function set is closed under linear combination and polynomial extension, the composite kernels are also valid. The reason we propose to use a feature-based kernel is that we can have a clear idea of what syntactic clues it represents and what kind of information it misses. This is important when developing or refining kernels, so that we can make them generate complementary information from different syntactic processing results.

5 Experiments

Experiments were carried out on the ACE RDR (Relation Detection and Recognition) task using hand-annotated entities, provided as part of the ACE evaluation. The ACE corpora contain documents from two sources: newswire (nwire) documents and broadcast news transcripts (bnews). In this section we will compare performance of different kernel setups trained with SVM, as well as different classifiers, KNN and SVM, with the same kernel setup. The SVM package we used is SVM^{light}. The training parameters were chosen using cross-validation. One-against-all classification was applied to each pair of entities in a sentence. When SVM predictions conflict on a relation example, the one with larger margin will be selected as the final answer.

5.1 Corpus

The ACE RDR training data contains 348 documents, 125K words and 4400 relations. It consists of both nwire and bnews documents. Evaluation of kernels was done on the training data using 5-fold cross-validation. We also evaluated the full kernel setup with SVM on the official test data, which is about half the size of the training data. All the data is preprocessed by the Charniak parser and GLARF dependency analyzer. Then relation examples are generated based these results.

5.2 Results

Table 2 shows the performance of the SVM on different kernel setups. The kernel setups in this experiment are incremental. From this table we can see that adding kernels continuously improves the performance, which indicates they provide additional clues to the previous setup. The argument kernel treats the two arguments as independent entities. The link sequence kernel introduces the syntactic connection between arguments, so adding it to the argument kernel boosted the performance. Setup F shows the performance of adding only dependency kernels to the argument kernel. The performance is not as good as setup B, indicating that dependency information alone is not as crucial as the link sequence.

	Kernel	Performance		
		prec	recall	F-score
A	Argument *(Ψ_1)*	52.96%	58.47%	55.58%
B	***A + link** (Ψ_1+Ψ_3)*	58.77%	71.25%	64.41%*
C	***B-poly** (Φ_1)*	66.98%	70.33%	68.61%*
D	***C + dep** (Φ_1+Ψ_4+Ψ_5)*	69.10%	71.41%	70.23%*
E	***D + bigram** (Φ_2)*	69.23%	70.50%	<u>70.35%</u>
F	***A + dep** (Ψ_1+Ψ_4+Ψ_5)*	57.86%	68.50%	62.73%

Table 2. SVM performance on incremental kernel setups. Each setup adds one level of kernels to the previous one except setup F. Evaluated on the ACE training data with 5-fold cross-validation. F-scores marked by * are significantly better than the previous setup (at 95% confidence level).

Another observation is that adding the bigram kernel in the presence of all other level of kernels improved both precision and recall, indicating that it helped in both correcting errors in other processing results and providing supplementary information missed by other levels of analysis. In another experiment evaluated on the nwire data only (about half of the training data), adding the bigram kernel improved F-score 0.5% and this improvement is statistically significant.

Type	KNN (Ψ_1+Ψ_3)	KNN (Φ_2)	SVM (Φ_2)
EMP-ORG	75.43%	72.66%	77.76%
PHYS	62.19 %	61.97%	66.37%
GPE-AFF	58.67%	56.22%	62.13%
PER-SOC	65.11%	65.61%	73.46%
DISC	68.20%	62.91%	66.24%
ART	69.59%	68.65%	67.68%
Other-AFF	51.05%	55.20%	46.55%
Total	67.44%	65.69%	70.35%

Table 3. Performance of SVM and KNN (k=3) on different kernel setups. Types are ordered in decreasing order of frequency of occurrence in the ACE corpus. In SVM training, the same parameters were used for all 7 types.

Table 3 shows the performance of SVM and KNN (k Nearest Neighbors) on different kernel setups. For KNN, k was set to 3. In the first setup of KNN, the two kernels which seem to contain most of the important information are used. It performs quite well when compared with the SVM result. The other two tests are based on the full kernel setup. For the two KNN experiments, adding more kernels (features) does not help. The reason might be that all kernels (features) were weighted equally in the composite kernel Φ_2 and this may not be optimal for KNN. Another reason is that the polynomial extension of kernels does not have any benefit in KNN because it is a monotonic transformation of similarity values. So the results of KNN on kernel $(\Psi_1+\Psi_3)$ and Φ_1 would be exactly the same. We also tried different k for KNN and k=3 seems to be the best choice in either case.

For the four major types of relations SVM does better than KNN, probably due to SVM's generalization ability in the presence of large numbers of features. For the last three types with many fewer examples, performance of SVM is not as good as KNN. The reason we think is that training of SVM on these types is not sufficient. We tried different training parameters for the types with fewer examples, but no dramatic improvement obtained.

We also evaluated our approach on the official ACE RDR test data and obtained very competitive scores.[3] The primary scoring metric[4] for the ACE evaluation is a 'value' score, which is computed by deducting from 100 a penalty for each missing and spurious relation; the penalty depends on the types of the arguments to the relation. The value scores produced by the ACE scorer for nwire and bnews test data are 71.7 and 68.0 repectively. The value score on all data is 70.1.[5] The scorer also reports an F-score based on full or partial match of relations to the keys. The unweighted F-score for this test produced by the ACE scorer on all data is 76.0%. For this evaluation we used nearest neighbor to determine argument ordering and relation subtypes.

The classification scheme in our experiments is one-against-all. It turned out there is not so much confusion between relation types. The confusion matrix of predictions is fairly clean. We also tried pairwise classification, and it did not help much.

6 Discussion

In this paper, we have shown that using kernels to combine information from different syntactic sources performed well on the entity relation detection task. Our experiments show that each level of syntactic processing contains useful information for the task. Combining them may provide complementary information to overcome errors arising from linguistic analysis. Especially, low level information obtained with high reliability helped with the other deep processing results. This design feature of our approach should be best employed when the preprocessing errors at each level are independent, namely when there is no dependency between the preprocessing modules. The model was tested on text with annotated entities, but its design is generic. It can work with

[3] As ACE participants, we are bound by the participation agreement not to disclose other sites' scores, so no direct comparison can be provided.

[4] http://www.nist.gov/speech/tests/ace/ace04/software.htm

[5] No comparable inter-annotator agreement scores are available for this task, with pre-defined entities. However, the agreement scores across multiple sites for similar relation tagging tasks done in early 2005, using the value metric, ranged from about 0.70 to 0.80.

noisy entity detection input from an automatic tagger. With all the existing information from other processing levels, this model can be also expected to recover from errors in entity tagging.

7 Further Work

Kernel functions have many nice properties. There are also many well known kernels, such as radial basis kernels, which have proven successful in other areas. In the work described here, only linear combinations and polynomial extensions of kernels have been evaluated. We can explore other kernel properties to integrate the existing syntactic kernels. In another direction, training data is often sparse for IE tasks. String matching is not sufficient to capture semantic similarity of words. One solution is to use general purpose corpora to create clusters of similar words; another option is to use available resources like WordNet. These word similarities can be readily incorporated into the kernel framework. To deal with sparse data, we can also use deeper text analysis to capture more regularities from the data. Such analysis may be based on newly-annotated corpora like PropBank (Kingsbury and Palmer, 2002) at the University of Pennsylvania and NomBank (Meyers et al., 2004) at New York University. Analyzers based on these resources can generate regularized semantic representations for lexically or syntactically related sentence structures. Although deeper analysis may even be less accurate, our framework is designed to handle this and still obtain some improvement in performance.

8 Acknowledgement

This research was supported in part by the Defense Advanced Research Projects Agency under Grant N66001-04-1-8920 from SPAWAR San Diego, and by the National Science Foundation under Grant ITS-0325657. This paper does not necessarily reflect the position of the U.S. Government. We wish to thank Adam Meyers of the NYU NLP group for his help in producing deep dependency analyses.

References

M. Collins and S. Miller. 1997. *Semantic tagging using a probabilistic context free grammar*. In Proceedings of the 6th Workshop on Very Large Corpora.

N. Cristianini and J. Shawe-Taylor. 2000. *An introduction to support vector machines*. Cambridge University Press.

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.

D. Gildea and M. Palmer. 2002. *The Necessity of Parsing for Predicate Argument Recognition*. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.

N. Kambhatla. 2004. *Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations*. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics.

P. Kingsbury and M. Palmer. 2002. *From treebank to propbank*. In Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002).

C. D. Manning and H. Schutze 2002. Foundations of *Statistical Natural Language Processing.* The MIT Press, page 454-455.

A. Meyers, R. Grishman, M. Kosaka and S. Zhao. 2001. *Covering Treebanks with GLARF*. In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics.

A. Meyers, R. Reeves, Catherine Macleod, Rachel Szekeley, Veronkia Zielinska, Brian Young, and R. Grishman. 2004. *The Cross-Breeding of Dictionaries*. In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2004).

S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. 2000. *A novel use of statistical parsing to extract information from text*. In 6th Applied Natural Language Processing Conference.

K.-R. Müller, S. Mika, G. Ratsch, K. Tsuda and B. Scholkopf. 2001. *An introduction to kernel-based learning algorithms*, IEEE Trans. Neural Networks, 12, 2, pages 181-201.

V. N. Vapnik. 1998. *Statistical Learning Theory.* Wiley-Interscience Publication.

D. Zelenko, C. Aone and A. Richardella. 2003. *Kernel methods for relation extraction*. Journal of Machine Learning Research.

Shubin Zhao, Adam Meyers, Ralph Grishman. 2004. *Discriminative Slot Detection Using Kernel Methods*. In the Proceedings of the 20th International Conference on Computational Linguistics.

Exploring Various Knowledge in Relation Extraction

ZHOU GuoDong SU Jian ZHANG Jie ZHANG Min
Institute for Infocomm research
21 Heng Mui Keng Terrace, Singapore 119613
Email: {zhougd, sujian, zhangjie, mzhang}@i2r.a-star.edu.sg

Abstract

Extracting semantic relationships between entities is challenging. This paper investigates the incorporation of diverse lexical, syntactic and semantic knowledge in feature-based relation extraction using SVM. Our study illustrates that the base phrase chunking information is very effective for relation extraction and contributes to most of the performance improvement from syntactic aspect while additional information from full parsing gives limited further enhancement. This suggests that most of useful information in full parse trees for relation extraction is shallow and can be captured by chunking. We also demonstrate how semantic information such as WordNet and Name List, can be used in feature-based relation extraction to further improve the performance. Evaluation on the ACE corpus shows that effective incorporation of diverse features enables our system outperform previously best-reported systems on the 24 ACE relation subtypes and significantly outperforms tree kernel-based systems by over 20 in F-measure on the 5 ACE relation types.

1 Introduction

With the dramatic increase in the amount of textual information available in digital archives and the WWW, there has been growing interest in techniques for automatically extracting information from text. Information Extraction (IE) systems are expected to identify relevant information (usually of pre-defined types) from text documents in a certain domain and put them in a structured format.

According to the scope of the NIST Automatic Content Extraction (ACE) program, current research in IE has three main objectives: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC). The EDT task entails the detection of entity mentions and chaining them together by identifying their coreference. In ACE vocabulary, entities are objects, mentions are references to them, and relations are semantic relationships between entities. Entities can be of five types: persons, organizations, locations, facilities and geo-political entities (GPE: geographically defined regions that indicate a political boundary, e.g. countries, states, cities, etc.). Mentions have three levels: names, nomial expressions or pronouns. The RDC task detects and classifies implicit and explicit relations[1] between entities identified by the EDT task. For example, we want to determine whether a person is at a location, based on the evidence in the context. Extraction of semantic relationships between entities can be very useful for applications such as question answering, e.g. to answer the query "Who is the president of the United States?".

This paper focuses on the ACE RDC task and employs diverse lexical, syntactic and semantic knowledge in feature-based relation extraction using Support Vector Machines (SVMs). Our study illustrates that the base phrase chunking information contributes to most of the performance inprovement from syntactic aspect while additional full parsing information does not contribute much, largely due to the fact that most of relations defined in ACE corpus are within a very short distance. We also demonstrate how semantic information such as WordNet (Miller 1990) and Name List can be used in the feature-based framework. Evaluation shows that the incorporation of diverse features enables our system achieve best reported performance. It also shows that our fea-

[1] In ACE (http://www.ldc.upenn.edu/Projects/ACE), explicit relations occur in text with explicit evidence suggesting the relationships. Implicit relations need not have explicit supporting evidence in text, though they should be evident from a reading of the document.

ture-based approach outperforms tree kernel-based approaches by 11 F-measure in relation detection and more than 20 F-measure in relation detection and classification on the 5 ACE relation types.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 and Section 4 describe our approach and various features employed respectively. Finally, we present experimental setting and results in Section 5 and conclude with some general observations in relation extraction in Section 6.

2 Related Work

The relation extraction task was formulated at the 7th Message Understanding Conference (MUC-7 1998) and is starting to be addressed more and more within the natural language processing and machine learning communities.

Miller et al (2000) augmented syntactic full parse trees with semantic information corresponding to entities and relations, and built generative models for the augmented trees. Zelenko et al (2003) proposed extracting relations by computing kernel functions between parse trees. Culotta et al (2004) extended this work to estimate kernel functions between augmented dependency trees and achieved 63.2 F-measure in relation detection and 45.8 F-measure in relation detection and classification on the 5 ACE relation types. Kambhatla (2004) employed Maximum Entropy models for relation extraction with features derived from word, entity type, mention level, overlap, dependency tree and parse tree. It achieves 52.8 F-measure on the 24 ACE relation subtypes. Zhang (2004) approached relation classification by combining various lexical and syntactic features with bootstrapping on top of Support Vector Machines.

Tree kernel-based approaches proposed by Zelenko et al (2003) and Culotta et al (2004) are able to explore the implicit feature space without much feature engineering. Yet further research work is still expected to make it effective with complicated relation extraction tasks such as the one defined in ACE. Complicated relation extraction tasks may also impose a big challenge to the modeling approach used by Miller et al (2000) which integrates various tasks such as part-of-speech tagging, named entity recognition, template element extraction and relation extraction, in a single model.

This paper will further explore the feature-based approach with a systematic study on the extensive incorporation of diverse lexical, syntactic and semantic information. Compared with Kambhatla (2004), we separately incorporate the base phrase chunking information, which contributes to most of the performance improvement from syntactic aspect. We also show how semantic information like WordNet and Name List can be equipped to further improve the performance. Evaluation on the ACE corpus shows that our system outperforms Kambhatla (2004) by about 3 F-measure on extracting 24 ACE relation subtypes. It also shows that our system outperforms tree kernel-based systems (Culotta et al 2004) by over 20 F-measure on extracting 5 ACE relation types.

3 Support Vector Machines

Support Vector Machines (SVMs) are a supervised machine learning technique motivated by the statistical learning theory (Vapnik 1998). Based on the structural risk minimization of the statistical learning theory, SVMs seek an optimal separating hyper-plane to divide the training examples into two classes and make decisions based on support vectors which are selected as the only effective instances in the training set.

Basically, SVMs are binary classifiers. Therefore, we must extend SVMs to multi-class (e.g. K) such as the ACE RDC task. For efficiency, we apply the *one vs. others* strategy, which builds K classifiers so as to separate one class from all others, instead of the *pairwise* strategy, which builds K*(K-1)/2 classifiers considering all pairs of classes. The final decision of an instance in the multiple binary classification is determined by the class which has the maximal SVM output. Moreover, we only apply the simple linear kernel, although other kernels can peform better.

The reason why we choose SVMs for this purpose is that SVMs represent the state-of–the-art in the machine learning research community, and there are good implementations of the algorithm available. In this paper, we use the binary-class SVMLight[2] deleveloped by Joachims (1998).

[2] Joachims has just released a new version of SVMLight for multi-class classification. However, this paper only uses the binary-class version. For details about SVMLight, please see http://svmlight.joachims.org/

4 Features

The semantic relation is determined between two mentions. In addition, we distinguish the argument order of the two mentions (M1 for the first mention and M2 for the second mention), e.g. M1-Parent-Of-M2 vs. M2-Parent-Of-M1. For each pair of mentions3, we compute various lexical, syntactic and semantic features.

4.1 Words

According to their positions, four categories of words are considered: 1) the words of both the mentions, 2) the words between the two mentions, 3) the words before M1, and 4) the words after M2. For the words of both the mentions, we also differentiate the head word[4] of a mention from other words since the head word is generally much more important. The words between the two mentions are classified into three bins: the first word in between, the last word in between and other words in between. Both the words before M1 and after M2 are classified into two bins: the first word next to the mention and the second word next to the mention. Since a pronominal mention (especially neutral pronoun such as 'it' and 'its') contains little information about the sense of the mention, the coreference chain is used to decide its sense. This is done by replacing the pronominal mention with the most recent non-pronominal antecedent when determining the word features, which include:

- WM1: bag-of-words in M1
- HM1: head word of M1
- WM2: bag-of-words in M2
- HM2: head word of M2
- HM12: combination of HM1 and HM2
- WBNULL: when no word in between
- WBFL: the only word in between when only one word in between
- WBF: first word in between when at least two words in between
- WBL: last word in between when at least two words in between
- WBO: other words in between except first and last words when at least three words in between
- BM1F: first word before M1
- BM1L: second word before M1
- AM2F: first word after M2
- AM2L: second word after M2

4.2 Entity Type

This feature concerns about the entity type of both the mentions, which can be PERSON, ORGANIZATION, FACILITY, LOCATION and Geo-Political Entity or GPE:

- ET12: combination of mention entity types

4.3 Mention Level

This feature considers the entity level of both the mentions, which can be NAME, NOMIAL and PRONOUN:

- ML12: combination of mention levels

4.4 Overlap

This category of features includes:

- #MB: number of other mentions in between
- #WB: number of words in between
- M1>M2 or M1<M2: flag indicating whether M2/M1is included in M1/M2.

Normally, the above overlap features are too general to be effective alone. Therefore, they are also combined with other features: 1) ET12+M1>M2; 2) ET12+M1<M2; 3) HM12+M1>M2; 4) HM12+M1<M2.

4.5 Base Phrase Chunking

It is well known that chunking plays a critical role in the Template Relation task of the 7th Message Understanding Conference (MUC-7 1998). The related work mentioned in Section 2 extended to explore the information embedded in the full parse trees. In this paper, we separate the features of base

[3] In ACE, each mention has a head annotation and an extent annotation. In all our experimentation, we only consider the word string between the beginning point of the extent annotation and the end point of the head annotation. This has an effect of choosing the base phrase contained in the extent annotation. In addition, this also can reduce noises without losing much of information in the mention. For example, in the case where the noun phrase "the former CEO of McDonald" has the head annotation of "CEO" and the extent annotation of "the former CEO of McDonald", we only consider "the former CEO" in this paper.

[4] In this paper, the head word of a mention is normally set as the last word of the mention. However, when a preposition exists in the mention, its head word is set as the last word before the preposition. For example, the head word of the name mention "University of Michigan" is "University".

phrase chunking from those of full parsing. In this way, we can separately evaluate the contributions of base phrase chunking and full parsing. Here, the base phrase chunks are derived from full parse trees using the Perl script[5] written by Sabine Buchholz from Tilburg University and the Collins' parser (Collins 1999) is employed for full parsing. Most of the chunking features concern about the head words of the phrases between the two mentions. Similar to word features, three categories of phrase heads are considered: 1) the phrase heads in between are also classified into three bins: the first phrase head in between, the last phrase head in between and other phrase heads in between; 2) the phrase heads before M1 are classified into two bins: the first phrase head before and the second phrase head before; 3) the phrase heads after M2 are classified into two bins: the first phrase head after and the second phrase head after. Moreover, we also consider the phrase path in between.

- CPHBNULL when no phrase in between
- CPHBFL: the only phrase head when only one phrase in between
- CPHBF: first phrase head in between when at least two phrases in between
- CPHBL: last phrase head in between when at least two phrase heads in between
- CPHBO: other phrase heads in between except first and last phrase heads when at least three phrases in between
- CPHBM1F: first phrase head before M1
- CPHBM1L: second phrase head before M1
- CPHAM2F: first phrase head after M2
- CPHAM2F: second phrase head after M2
- CPP: path of phrase labels connecting the two mentions in the chunking
- CPPH: path of phrase labels connecting the two mentions in the chunking augmented with head words, if at most two phrases in between

4.6 Dependency Tree

This category of features includes information about the words, part-of-speeches and phrase labels of the words on which the mentions are dependent in the dependency tree derived from the syntactic full parse tree. The dependency tree is built by using the phrase head information returned by the Collins' parser and linking all the other fragments in a phrase to its head. It also includes flags indicating whether the two mentions are in the same NP/PP/VP.

- ET1DW1: combination of the entity type and the dependent word for M1
- H1DW1: combination of the head word and the dependent word for M1
- ET2DW2: combination of the entity type and the dependent word for M2
- H2DW2: combination of the head word and the dependent word for M2
- ET12SameNP: combination of ET12 and whether M1 and M2 included in the same NP
- ET12SamePP: combination of ET12 and whether M1 and M2 exist in the same PP
- ET12SameVP: combination of ET12 and whether M1 and M2 included in the same VP

4.7 Parse Tree

This category of features concerns about the information inherent only in the full parse tree.

- PTP: path of phrase labels (removing duplicates) connecting M1 and M2 in the parse tree
- PTPH: path of phrase labels (removing duplicates) connecting M1 and M2 in the parse tree augmented with the head word of the top phrase in the path.

4.8 Semantic Resources

Semantic information from various resources, such as WordNet, is used to classify important words into different semantic lists according to their indicating relationships.

Country Name List

This is to differentiate the relation subtype "ROLE.Citizen-Of", which defines the relationship between a person and the country of the person's citizenship, from other subtypes, especially "ROLE.Residence", where defines the relationship between a person and the location in which the person lives. Two features are defined to include this information:

- ET1Country: the entity type of M1 when M2 is a country name
- CountryET2: the entity type of M2 when M1 is a country name

[5] http://ilk.kub.nl/~sabine/chunklink/

Personal Relative Trigger Word List

This is used to differentiate the six personal social relation subtypes in ACE: Parent, Grandparent, Spouse, Sibling, Other-Relative and Other-Personal. This trigger word list is first gathered from WordNet by checking whether a word has the semantic class "person|…|relative". Then, all the trigger words are semi-automatically[6] classified into different categories according to their related personal social relation subtypes. We also extend the list by collecting the trigger words from the head words of the mentions in the training data according to their indicating relationships. Two features are defined to include this information:

- ET1SC2: combination of the entity type of M1 and the semantic class of M2 when M2 triggers a personal social subtype.
- SC1ET2: combination of the entity type of M2 and the semantic class of M1 when the first mention triggers a personal social subtype.

5 Experimentation

This paper uses the ACE corpus provided by LDC to train and evaluate our feature-based relation extraction system. The ACE corpus is gathered from various newspapers, newswire and broadcasts. In this paper, we only model explicit relations because of poor inter-annotator agreement in the annotation of implicit relations and their limited number.

5.1 Experimental Setting

We use the official ACE corpus from LDC. The training set consists of 674 annotated text documents (~300k words) and 9683 instances of relations. During development, 155 of 674 documents in the training set are set aside for fine-tuning the system. The testing set is held out only for final evaluation. It consists of 97 documents (~50k words) and 1386 instances of relations. Table 1 lists the types and subtypes of relations for the ACE Relation Detection and Characterization (RDC) task, along with their frequency of occurrence in the ACE training set. It shows that the ACE corpus suffers from a small amount of annotated data for a few subtypes such as the subtype "Founder" under the type "ROLE". It also shows that the ACE RDC task defines some difficult subtypes such as the subtypes "Based-In", "Located" and "Residence" under the type "AT", which are difficult even for human experts to differentiate.

Type	Subtype	Freq
AT(2781)	Based-In	347
	Located	2126
	Residence	308
NEAR(201)	Relative-Location	201
PART(1298)	Part-Of	947
	Subsidiary	355
	Other	6
ROLE(4756)	Affiliate-Partner	204
	Citizen-Of	328
	Client	144
	Founder	26
	General-Staff	1331
	Management	1242
	Member	1091
	Owner	232
	Other	158
SOCIAL(827)	Associate	91
	Grandparent	12
	Other-Personal	85
	Other-Professional	339
	Other-Relative	78
	Parent	127
	Sibling	18
	Spouse	77

Table 1: Relation types and subtypes in the ACE training data

In this paper, we explicitly model the argument order of the two mentions involved. For example, when comparing mentions m1 and m2, we distinguish between m1-ROLE.Citizen-Of-m2 and m2-ROLE.Citizen-Of-m1. Note that only 6 of these 24 relation subtypes are symmetric: "Relative-Location", "Associate", "Other-Relative", "Other-Professional", "Sibling", and "Spouse". In this way, we model relation extraction as a multi-class classification problem with 43 classes, two for each relation subtype (except the above 6 symmetric subtypes) and a "NONE" class for the case where the two mentions are not related.

5.2 Experimental Results

In this paper, we only measure the performance of relation extraction on "true" mentions with "true" chaining of coreference (i.e. as annotated by the corpus annotators) in the ACE corpus. Table 2 measures the performance of our relation extrac-

[6] Those words that have the semantic classes "Parent", "GrandParent", "Spouse" and "Sibling" are automatically set with the same classes without change. However, The remaining words that do not have above four classes are manually classified.

tion system over the 43 ACE relation subtypes on the testing set. It shows that our system achieves best performance of 63.1%/49.5%/ 55.5 in precision/recall/F-measure when combining diverse lexical, syntactic and semantic features. Table 2 also measures the contributions of different features by gradually increasing the feature set. It shows that:

Features	P	R	F
Words	69.2	23.7	35.3
+Entity Type	67.1	32.1	43.4
+Mention Level	67.1	33.0	44.2
+Overlap	57.4	40.9	47.8
+Chunking	61.5	46.5	53.0
+Dependency Tree	62.1	47.2	53.6
+Parse Tree	62.3	47.6	54.0
+Semantic Resources	63.1	49.5	55.5

Table 2: Contribution of different features over 43 relation subtypes in the test data

- Using word features only achieves the performance of 69.2%/23.7%/35.3 in precision/recall/F-measure.
- Entity type features are very useful and improve the F-measure by 8.1 largely due to the recall increase.
- The usefulness of mention level features is quite limited. It only improves the F-measure by 0.8 due to the recall increase.
- Incorporating the overlap features gives some balance between precision and recall. It increases the F-measure by 3.6 with a big precision decrease and a big recall increase.
- Chunking features are very useful. It increases the precision/recall/F-measure by 4.1%/5.6%/ 5.2 respectively.
- To our surprise, incorporating the dependency tree and parse tree features only improve the F-measure by 0.6 and 0.4 respectively. This may be due to the fact that most of relations in the ACE corpus are quite local. Table 3 shows that about 70% of relations exist where two mentions are embedded in each other or separated by at most one word. While short-distance relations dominate and can be resolved by above simple features, the dependency tree and parse tree features can only take effect in the remaining much less long-distance relations. However, full parsing is always prone to long distance errors although the Collins' parser used in our system represents the state-of-the-art in full parsing.
- Incorporating semantic resources such as the country name list and the personal relative trigger word list further increases the F-measure by 1.5 largely due to the differentiation of the relation subtype "ROLE.Citizen-Of" from "ROLE. Residence" by distinguishing country GPEs from other GPEs. The effect of personal relative trigger words is very limited due to the limited number of testing instances over personal social relation subtypes.

Table 4 separately measures the performance of different relation types and major subtypes. It also indicates the number of testing instances, the number of correctly classified instances and the number of wrongly classified instances for each type or subtype. It is not surprising that the performance on the relation type "NEAR" is low because it occurs rarely in both the training and testing data. Others like "PART.Subsidary" and "SOCIAL. Other-Professional" also suffer from their low occurrences. It also shows that our system performs best on the subtype "SOCIAL.Parent" and "ROLE. Citizen-Of". This is largely due to incorporation of two semantic resources, i.e. the country name list and the personal relative trigger word list. Table 4 also indicates the low performance on the relation type "AT" although it frequently occurs in both the training and testing data. This suggests the difficulty of detecting and classifying the relation type "AT" and its subtypes.

Table 5 separates the performance of relation detection from overall performance on the testing set. It shows that our system achieves the performance of 84.8%/66.7%/74.7 in precision/recall/F-measure on relation detection. It also shows that our system achieves overall performance of 77.2%/60.7%/68.0 and 63.1%/49.5%/55.5 in precision/recall/F-measure on the 5 ACE relation types and the best-reported systems on the ACE corpus. It shows that our system achieves better performance by ~3 F-measure largely due to its gain in recall. It also shows that feature-based methods dramatically outperform kernel methods. This suggests that feature-based methods can effectively combine different features from a variety of sources (e.g. WordNet and gazetteers) that can be brought to bear on relation extraction. The tree kernels developed in Culotta et al (2004) are yet to be effective on the ACE RDC task.

Finally, Table 6 shows the distributions of errors. It shows that 73% (627/864) of errors results from relation detection and 27% (237/864) of errors results from relation characterization, among which 17.8% (154/864) of errors are from misclassification across relation types and 9.6% (83/864) of errors are from misclassification of relation subtypes inside the same relation types. This suggests that relation detection is critical for relation extraction.

# of relations		# of other mentions in between					
		0	1	2	3	>=4	Overall
#	0	3991	161	11	0	0	4163
of	1	2350	315	26	2	0	2693
the words	2	465	95	7	2	0	569
in	3	311	234	14	0	0	559
between	4	204	225	29	2	3	463
	5	111	113	38	2	1	265
	>=6	262	297	277	148	134	1118
	Overall	7694	1440	402	156	138	**9830**

Table 3: Distribution of relations over #words and #other mentions in between in the training data

Type	Subtype	#Testing Instances	#Correct	#Error	P	R	F
AT		**392**	**224**	**105**	**68.1**	**57.1**	**62.1**
	Based-In	85	39	10	79.6	45.9	58.2
	Located	241	132	120	52.4	54.8	53.5
	Residence	66	19	9	67.9	28.8	40.4
NEAR		**35**	**8**	**1**	**88.9**	**22.9**	**36.4**
	Relative-Location	35	8	1	88.9	22.9	36.4
PART		**164**	**106**	**39**	**73.1**	**64.6**	**68.6**
	Part-Of	136	76	32	70.4	55.9	62.3
	Subsidiary	27	14	23	37.8	51.9	43.8
ROLE		**699**	**443**	**82**	**84.4**	**63.4**	**72.4**
	Citizen-Of	36	25	8	75.8	69.4	72.6
	General-Staff	201	108	46	71.1	53.7	62.3
	Management	165	106	72	59.6	64.2	61.8
	Member	224	104	36	74.3	46.4	57.1
SOCIAL		**95**	**60**	**21**	**74.1**	**63.2**	**68.5**
	Other-Professional	29	16	32	33.3	55.2	41.6
	Parent	25	17	0	100	68.0	81.0

Table 4: Performance of different relation types and major subtypes in the test data

System	Relation Detection			RDC on Types			RDC on Subtypes		
	P	R	F	P	R	F	P	R	F
Ours: feature-based	**84.8**	**66.7**	**74.7**	**77.2**	**60.7**	**68.0**	63.1	**49.5**	**55.5**
Kambhatla (2004):feature-based	-	-	-	-	-	-	**63.5**	45.2	52.8
Culotta et al (2004):tree kernel	81.2	51.8	63.2	67.1	35.0	45.8	-	-	-

Table 5: Comparison of our system with other best-reported systems on the ACE corpus

Error Type		#Errors
Detection Error	False Negative	462
	False Positive	165
Characterization Error	Cross Type Error	154
	Inside Type Error	83

Table 6: Distribution of errors

6 Discussion and Conclusion

In this paper, we have presented a feature-based approach for relation extraction where diverse lexical, syntactic and semantic knowledge are employed. Instead of exploring the full parse tree information directly as previous related work, we incorporate the base phrase chunking information first. Evaluation on the ACE corpus shows that base phrase chunking contributes to most of the performance improvement from syntactic aspect while further incorporation of the parse tree and dependence tree information only slightly improves the performance. This may be due to three reasons: First, most of relations defined in ACE have two mentions being close to each other. While short-distance relations dominate and can be resolved by simple features such as word and chunking features, the further dependency tree and parse tree features can only take effect in the remaining much less and more difficult long-distance relations. Second, it is well known that full parsing is always prone to long-distance parsing errors although the Collins' parser used in our system achieves the state-of-the-art performance. Therefore, the state-of-art full parsing still needs to be further enhanced to provide accurate enough information, especially PP (Preposition Phrase) attachment. Last, effective ways need to be explored to incorporate information embedded in the full parse trees. Besides, we also demonstrate how semantic information such as WordNet and Name List, can be used in feature-based relation extraction to further improve the performance.

The effective incorporation of diverse features enables our system outperform previously best-reported systems on the ACE corpus. Although tree kernel-based approaches facilitate the exploration of the implicit feature space with the parse tree structure, yet the current technologies are expected to be further advanced to be effective for relatively complicated relation extraction tasks such as the one defined in ACE where 5 types and 24 subtypes need to be extracted. Evaluation on the ACE RDC task shows that our approach of combining various kinds of evidence can scale better to problems, where we have a lot of relation types with a relatively small amount of annotated data. The experiment result also shows that our feature-based approach outperforms the tree kernel-based approaches by more than 20 F-measure on the extraction of 5 ACE relation types.

In the future work, we will focus on exploring more semantic knowledge in relation extraction, which has not been covered by current research. Moreover, our current work is done when the Entity Detection and Tracking (EDT) has been perfectly done. Therefore, it would be interesting to see how imperfect EDT affects the performance in relation extraction.

References

Agichtein E. and Gravano L. (2000). Snowball: Extracting relations from large plain text collections. In *Proceedings of 5th ACM International Conference on Digital Libraries*. 4-7 June 2000. San Antonio, TX.

Brin S. (1998). Extracting patterns and relations from the World Wide Web. In *Proceedings of WebDB workshop at 6th International Conference on Extending DataBase Technology (EDBT'1998).*23-27 March 1998, Valencia, Spain

Collins M. (1999). Head-driven statistical models for natural language parsing. *Ph.D. Dissertation*, University of Pennsylvania.

Collins M. and Duffy N. (2002). Covolution kernels for natural language. In Dietterich T.G., Becker S. and Ghahramani Z. editors. *Advances in Neural Information Processing Systems 14*. Cambridge, MA.

Culotta A. and Sorensen J. (2004). Dependency tree kernels for relation extraction. In *Proceedings of 42th Annual Meeting of the Association for Computational Linguistics*. 21-26 July 2004. Barcelona, Spain

Cumby C.M. and Roth D. (2003). On kernel methods for relation learning. In Fawcett T. and Mishra N. editors. In Proceedings of 20th International Conference on Machine Learning (ICML'2003). 21-24 Aug 2003. Washington D.C. USA. AAAI Press.

Haussler D. (1999). Covention kernels on discrete structures. *Technical Report UCS-CRL-99-10*. University of California, Santa Cruz.

Joachims T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. In *Proceedings of European Conference on Machine Learning(ECML'1998)*. 21-23 April 1998. Chemnitz, Germany

Miller G.A. (1990). WordNet: An online lexical database. *International Journal of Lexicography*. 3(4):235-312.

Miller S., Fox H., Ramshaw L. and Weischedel R. (2000). A novel use of statistical parsing to extract information from text. In *Proceedings of 6th Applied Natural Language Processing Conference*. 29 April - 4 May 2000, Seattle, USA

MUC-7. (1998). *Proceedings of the 7th Message Understanding Conference (MUC-7).* Morgan Kaufmann, San Mateo, CA.

Kambhatla N. (2004). Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations. In *Proceedings of 42th Annual Meeting of the Association for Computational Linguistics*. 21-26 July 2004. Barcelona, Spain.

Roth D. and Yih W.T. (2002). Probabilistic reasoning for entities and relation recognition. In *Proceedings of 19th International Conference on Computational Linguistics(CoLING'2002)*. Taiwan.

Vapnik V. (1998). Statistical Learning Theory. Whiley, Chichester, GB.

Zelenko D., Aone C. and Richardella. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*. pp1083-1106.

Zhang Z. (2004). Weekly-supervised relation classification for Information Extraction. In Proceedings of ACM 13th Conference on Information and Knowledge Management (CIKM'2004). 8-13 Nov 2004. Washington D.C., USA.

A Quantitative Analysis of Lexical Differences Between Genders in Telephone Conversations

Constantinos Boulis
Department of Electrical Engineering
University of Washington
Seattle, 98195
boulis@ee.washington.edu

Mari Ostendorf
Department of Electrical Engineering
University of Washington
Seattle, 98195
mo@ee.washington.edu

Abstract

In this work, we provide an empirical analysis of differences in word use between genders in telephone conversations, which complements the considerable body of work in sociolinguistics concerned with gender linguistic differences. Experiments are performed on a large speech corpus of roughly 12000 conversations. We employ machine learning techniques to automatically categorize the gender of each speaker given only the transcript of his/her speech, achieving 92% accuracy. An analysis of the most characteristic words for each gender is also presented. Experiments reveal that the gender of one conversation side influences lexical use of the other side. A surprising result is that we were able to classify male-only vs. female-only conversations with almost perfect accuracy.

1 Introduction

Linguistic and prosodic differences between genders in American English have been studied for decades. The interest in analyzing the gender linguistic differences is two-fold. From the scientific perspective, it will increase our understanding of language production. From the engineering perspective, it can help improve the performance of a number of natural language processing tasks, such as text classification, machine translation or automatic speech recognition by training better language models. Traditionally, these differences have been investigated in the fields of sociolinguistics and psycholinguistics, see for example (Coates, 1997), (Eckert and McConnell-Ginet, 2003) or http://www.ling.lancs.ac.uk/groups/gal/genre.htm for a comprehensive bibliography on language and gender. Sociolinguists have approached the issue from a mostly non-computational perspective using relatively small and very focused data collections. Recently, the work of (Koppel et al., 2002) has used computational methods to characterize the differences between genders in written text, such as literary books. A number of monologues have been analyzed in (Singh, 2001) in terms of lexical richness using multivariate analysis techniques. The question of gender linguistic differences shares a number of issues with stylometry and author/speaker attribution research (Stamatatos et al., 2000), (Doddington, 2001), but novel issues emerge with analysis of conversational speech, such as studying the interaction of genders.

In this work, we focus on lexical differences between genders on telephone conversations and use machine learning techniques applied on text categorization and feature selection to characterize these differences. Therefore our conclusions are entirely data-driven. We use a very large corpus created for automatic speech recognition - the Fisher corpus described in (Cieri et al., 2004). The Fisher corpus is annotated with the gender of each speaker making it an ideal resource to study not only the characteristics of individual genders but also of gender pairs in spontaneous, conversational speech. The size and scope of the Fisher corpus is such that robust results can be derived for American English. The computational methods we apply can assist us in answering questions, such as *"To which degree are gender-discriminative words content-bearing words?"* or *"Which words are most characteristic for males in general or males talking to females?"*.

In section 2, we describe the corpus we have based our analysis on. In section 3, the machine learning tools are explained, while the experimental results are described in section 4 with a specific research question for each subsection. We conclude in section 5 with a summary and future directions.

2 The Corpus and Data Preparation

The Fisher corpus (Cieri et al., 2004) was used in all our experiments. It consists of telephone conversations between two people, randomly assigned to speak to each other. At the beginning of each conversation a topic is suggested at random from a list of 40. The latest release of the Fisher collection has more than 16 000 telephone conversations averaging 10 minutes each. Each person participates in 1-3 conversations, and each conversation is annotated with a topicality label. The topicality label gives the degree to which the suggested topic was followed and is an integer number from 0 to 4, 0 being the worse. In our site, we had an earlier version of the Fisher corpus with around 12 000 conversations. After removing conversations where at least one of the speakers was non-native[1] and conversations with topicality 0 or 1 we were left with 10 127 conversations. The original transcripts were minimally processed; acronyms were normalized to a sequence of characters with no intervening spaces, e.g. *t. v.* to *tv*; word fragments were converted to the same token *wordfragment*; all words were lowercased; and punctuation marks and special characters were removed. Some non-lexical tokens are maintained such as *laughter* and filled pauses such as *uh*, *um*. Backchannels and acknowledgments such as *uh-huh, mm-hmm* are also kept. The gender distribution of the Fisher corpus is 53% female and 47% male. Age distribution is 38% 16-29, 45% 30-49% and 17% 50+. Speakers were connected at random from a pool recruited in a national ad campaign. It is unlikely that the speakers knew their conversation partner. All major American English dialects are well represented, see (Cieri et al., 2004) for more details. The Fisher corpus was primarily created to facilitate automatic speech recognition research. The subset we have used has about 17.8M words or about 1 600 hours of speech and it is the largest resource ever used to analyze gender linguistic differences. In comparison, (Singh, 2001) has used about 30 000 words for their analysis.

[1]About 10% of speakers are non-native making this corpus suitable for investigating their lexical differences compared to American English speakers.

Before attempting to analyze the gender differences, there are two main biases that need to be removed. The first bias, which we term the *topic bias* is introduced by not accounting for the fact that the distribution of topics in males and females is uneven, despite the fact that the topic is pre-assigned randomly. For example, if topic A happened to be more common for males than females and we failed to account for that, then we would be implicitly building a topic classifier rather than a gender classifier. Our intention here is to analyze gender linguistic differences controlling for the topic effect as if both genders talk equally about the same topics. The second bias, which we term *speaker bias* is introduced by not accounting for the fact that specific speakers have idiosyncratic expressions. If our training data consisted of a small number of speakers appearing in both training and testing data, then we will be implicitly modeling speaker differences rather than gender differences.

To normalize for these two important biases, we made sure that both genders have the same percent of conversation sides for each topic and there are 8899 speakers in training and 2000 in testing with no overlap between the two sets. After these two steps, there were 14969 conversation sides used for training and 3738 sides for testing. The median length of a conversation side was 954.

3 Machine Learning Methods Used

The methods we have used for characterizing the differences between genders and gender pairs are similar to what has been used for the task of text classification. In text classification, the objective is to classify a document $\vec{d}$ to one (or more) of T predefined topics y. A number of N tuples $(\vec{d}_n, y_n)$

are provided for training the classifier. A major challenge of text classification is the very high dimensionality for representing each document which brings forward the need for feature selection, i.e. selecting the most discriminative words and discarding all others.

In this study, we chose two ways for characterizing the differences between gender categories. The first, is to classify the transcript of each speaker, i.e. each conversation side, to the appropriate gender category. This approach can show the cumulative effect of all terms on the distinctiveness of gender categories. The second approach is to apply feature selection methods, similar to those used in text categorization, to reveal the most characteristic features for each gender.

Classifying a transcript of speech according to gender can be done with a number of different learning methods. We have compared Support Vector Machines (SVMs), Naive Bayes, Maximum Entropy and the tfidf/Rocchio classifier and found SVMs to be the most successful. A possible difference between text classification and gender classification is that different methods for feature weighting may be appropriate. In text classification, inverse document frequency is applied to the frequency of each term resulting in the deweighting of common terms. This weighting scheme is effective for text classification because common terms do not contribute to the topic of a document. However, the reverse may be true for gender classification, where the common terms may be the ones that mostly contribute to the gender category. This is an issue that we will investigate in section 4 and has implications for the feature weighting scheme that needs to be applied to the vector representation.

In addition to classification, we have applied feature selection techniques to assess the discriminative ability of each individual feature. Information gain has been shown to be one of the most successful feature selection methods for text classification (Forman, 2003). It is given by:

$$IG(w) = H(\mathbf{C}) - p(w)H(\mathbf{C}|w) - p(\bar{w})H(\mathbf{C}|\bar{w}) \tag{1}$$

where $H(\mathbf{C}) = -\sum_{c=1}^{C} p(c) \log p(c)$ denotes the entropy of the discrete gender category random variable **C**. Each document is represented with the Bernoulli model, i.e. a vector of 1 or 0 depending if the word appears or not in the document. We have also implemented another feature selection mechanism, the KL-divergence, which is given by:

$$KL(w) = D[p(c|w)||p(c)] = \sum_{c=1}^{C} p(c|w) \log \frac{p(c|w)}{p(c)} \tag{2}$$

In the KL-divergence we have used the multinomial model, i.e. each document is represented as a vector of word counts. We smoothed the $p(w|c)$ distributions by assuming that every word in the vocabulary is observed at least 5 times for each class.

4 Experiments

Having explained the methods and data that we have used, we set forward to investigate a number of research questions concerning the nature of differences between genders. Each subsection is concerned with a single question.

4.1 Given only the transcript of a conversation, is it possible to classify conversation sides according to the gender of the speaker?

The first hypothesis we investigate is whether simple features, such as counts of individual terms (unigrams) or pairs of terms (bigrams) have different distributions between genders. The set of possible terms consists of all words in the Fisher corpus plus some non-lexical tokens such as laughter and filled pauses. One way to assess the difference in their distribution is by attempting to classify conversation sides according to the gender of the speaker. The results are shown in Table 1, where a number of different text classification algorithms were applied to classify conversation sides. 14969 conversation sides are used for training and 3738 sides are used for testing. No feature selection was performed; in all classifiers a vocabulary of all unigrams or bigrams with 5 or more occurrences is used (20513 for unigrams, 306779 for bigrams). For all algorithms, except Naive Bayes, we have used the tf·idf representation. The *Rainbow* toolkit (McCallum, 1996) was used for training the classifiers. Results show that differences between genders are clear and the best results are obtained by using SVMs. The fact that classification performance is significantly above chance for a variety of learning methods shows that lexical differences between genders are inherent in the data and not in a specific choice of classifier.

From Table 1 we also observe that using bigrams is consistently better than unigrams, despite the fact that the number of unique terms rises from ∼20K to ∼300K. This suggests that gender differences become even more profound for phrases, a finding similar to (Doddington, 2001) for speaker differences.

Table 1: Classification accuracy of different learning methods for the task of classifying the transcript of a conversation side according to the gender - male/female - of the speaker.

	Unigrams	Bigrams
Rocchio	76.3	86.5
Naive Bayes	83.0	89.2
MaxEnt	85.6	90.3
SVM	88.6	92.5

4.2 Does the gender of a conversation side influence lexical usage of the other conversation side?

Each conversation always consists of two people talking to each other. Up to this point, we have only attempted to analyze a conversation side in isolation, i.e. without using transcriptions from the other side. In this subsection, we attempt to assess the degree to which, if any, the gender of one speaker influences the language of the other speaker. In the first experiment, instead of defining two categories we define four; the Cartesian product of the gender of the current speaker and the gender of the other speaker. These categories are symbolized with two letters: the first characterizing the gender of the current speaker and the second the gender of the other speaker, i.e. FF, FM, MF, MM. The task remains the same: given the transcript of a conversation side, classify it according to the appropriate category. This is a task much harder than the binary classification we had in subsection 4.1, because given only the transcript of a conversation side we must make inferences about the gender of the current as well as the other conversation side. We have used SVMs as the learning method. In their basic formulation, SVMs are binary classifiers (although there has been recent work on multi-class SVMs). We followed the original binary formulation and converted the 4-class problem to 6 2-class problems. The final decision is taken by voting of the individual systems. The confusion matrix of the 4-way classification is shown in Table 2.

Table 2: Confusion matrix for 4-way classification of gender of both sides using transcripts from one side. Unigrams are used as features, SVMs as classification method. Each row represents the true category and each column the hypothesized category.

	FF	FM	MF	MM	F-measure
FF	1447	30	40	65	.778
FM	456	27	43	77	.074
MF	167	25	104	281	.214
MM	67	44	210	655	.638

The results show that although two of the four categories, FF and MM, are quite robustly detected the other two, FM and MF, are mostly confused with FF and MM respectively. These results can be mapped to single gender detection, giving accuracy of 85.9% for classifying the gender of the given transcript (as in Table 1) and 68.5% for classifying the gender of the conversational partner. The accuracy of 68.5% is higher than chance (57.8%) showing that genders alter their linguistic patterns depending on the gender of their conversational partner.

In the next experiment we design two binary classifiers. In the first classifier, the task is to correctly classify FF vs. MM transcripts, and in the second classifier the task is to classify FM vs. MF transcripts. Therefore, we attempt to classify the gender of a speaker given knowledge of whether the conversation is same-gender or cross-gender. For both classifiers 4526 sides were used for training equally divided among each class. 2558 sides were used for testing of the FF-MM classifier and 1180 sides for the FM-MF classifier. The results are shown in Table 3.

It is clear from Table 3 that there is a significant difference in performance between the FF-MM and FM-MF classifiers, suggesting that people alter their linguistic patterns depending on the gender of the person they are talking to. In same-gender conversations, almost perfect accuracy is reached, indicating that the linguistic patterns of the two genders be-

Table 3: Classification accuracies in same-gender and cross-gender conversations. SVMs are used as the classification method; no feature selection is applied.

	Unigrams	**Bigrams**
FF-MM	98.91	99.49
FM-MF	69.15	78.90

come very distinct. In cross-gender conversations the differences become less prominent since classification accuracy drops compared to same-gender conversations. This result, however, does not reveal how this convergence of linguistic patterns is achieved. Is it the case that the convergence is attributed to one of the genders, for example males attempting to match the patterns of females, or is it collectively constructed? To answer this question, we can examine the classification performance of two other binary classifiers FF vs. FM and MM vs. MF. The results are shown in Table 4. In both classifiers 4608 conversation sides are used for training, equally divided in each class. The number of sides used for testing is 989 and 689 for the FF-FM and MM-MF classifier respectively.

Table 4: Classifying the gender of speaker B given only the transcript of speaker A. SVMs are used as the classification method; no feature selection is applied.

	Unigrams	**Bigrams**
FF-FM	57.94	59.66
MM-MF	60.38	59.80

The results in Table 4 suggest that both genders equally alter their linguistic patterns to match the opposite gender. It is interesting to see that the gender of speaker B can be detected better than chance given only the transcript and gender of speaker A. The results are better than chance at the 0.0005 significance level.

4.3 Are some features more indicative of gender than other?

Having shown that gender lexical differences are prominent enough to classify each speaker according to gender quite robustly, another question is whether the high classification accuracies can be attributed to a small number of features or are rather the cumulative effect of a high number of them. In Table 5 we apply the two feature selection criteria that were described in 3.

Table 5: Effect of feature selection criteria on gender classification using SVM as the learning method. Horizontal axis refers to the fraction of the original vocabulary size (∼20K for unigrams, ∼300K for bigrams) that was used.

		1.0	**0.7**	**0.4**	**0.1**	**0.03**
KL	**1-gram**	88.6	88.8	87.8	86.3	85.6
	2-gram	92.5	92.6	92.2	91.9	90.3
IG	**1-gram**	88.6	88.5	88.9	87.6	87.0
	2-gram	92.5	92.4	92.6	91.8	90.8

The results of Table 5 show that lexical differences between genders are not isolated in a small set of words. The best results are achieved with 40% (IG) and 70% (KL) of the features, using fewer features steadily degrades the performance. Using the 5000 least discriminative unigrams and Naive Bayes as the classification method resulted in 58.4% classification accuracy which is not statistically better than chance (this is the test set of Tables 1 and 2 not of Table 4) . Using the 15000 least useful unigrams resulted in a classification accuracy of 66.4%, which shows that the number of irrelevant features is rather small, about 5K features.

It is also instructive to see which features are most discriminative for each gender. The features that when present are most indicative of each gender (positive features) are shown in Table 6. They are sorted using the KL distance and dropping the summation over both genders in equation (2). Looking at the top 2000 features for each number we observed that a number of swear words appear as most discriminative for males and family-relation terms are often associated with females. For example the following words are in the top 2000 (out of 20513) most useful features for males *shit, bullshit, shitty, fuck, fucking, fucked, bitching, bastards, ass, asshole, sucks, sucked, suck, sucker, damn, goddamn, damned*. The following words are in the top 2000 features for females *children, grandchild,*

Table 6: The 10 most discriminative features for each gender according to KL distance. Words higher in the list are more discriminative.

Male	Female
dude	*husband*
shit	*husband's*
fucking	*refunding*
wife	*goodness*
wife's	*boyfriend*
matt	*coupons*
steve	*crafts*
bass	*linda*
ben	*gosh*
fuck	*cute*

child, grandchildren, childhood, childbirth, kids, grandkids, son, grandson, daughter, granddaughter, boyfriend, marriage, mother, grandmother. It is also interesting to note that a number of non-lexical tokens are strongly associated with a certain gender. For example, *[laughter]* and acknowledgments/backchannels such as *uh-huh,uhuh* were in the top 2000 features for females. On the other hand, filled pauses such as *uh* were strong male indicators. Our analysis also reveals that a high number of useful features are names. A possible explanation is that people usually introduce themselves at the beginning of the conversation. In the top 30 words per gender, names represent over half of the words for males and nearly a quarter for females. Nearly a third were family-relations words for females, and 17

When examining cross-gender conversations, the discriminative words were quite substantially different. We can quantify the degree of change by measuring $KL_{SG}(w) - KL_{CG}(w)$ where $KL_{SG}(w)$ is the KL measure of word w for same-gender conversations. The analysis reveals that swear terms are highly associated with male-only conversations, while family-relation words are highly associated with female-only conversations.

From the traditional sociolinguistic perspective, these methods offer a way of discovering rather than testing words or phrases that have distinct usage between genders. For example, in a recent paper (Kiesling, in press) the word *dude* is analyzed as a male-to-male indicator. In our work, the word *dude* emerged as a male feature. As another example, our observation that some acknowledgments and backchannels (*uh-huh*) are more common for females than males while the reverse is true for filled pauses asserts a popular theory in sociolinguistics that males assume a more dominant role than females in conversations (Coates, 1997). Males tend to hold the floor more than women (more filled pauses) and females tend to be more responsive (more acknowledgments/backchannels).

4.4 Are gender-discriminative features content-bearing words?

Do the most gender-discriminative words contribute to the topic of the conversation, or are they simple fill-in words with no content? Since each conversation is labeled with one of 40 possible topics, we can rank features with IG or KL using topics instead of genders as categories. In fact, this is the standard way of performing feature selection for text classification. We can then compare the performance of classifying conversations to topics using the top-N features according to the gender or topic ranking. The results are shown in Table 7.

Table 7: Classification accuracies using topic- and gender-discriminative words, sorted using the information gain criterion. When randomly selecting 5000 features, 10 independent runs were performed and numbers reported are mean and standard deviation. Using the bottom 5000 topic words resulted in chance performance (∼5.0)

	Top 5K	**Bottom 5K**	**Random 5K**
Gender ranking	78.51	66.72	74.99±2.2
Topic ranking	87.72	-	74.99±2.2

From Table 7 we can observe that gender-discriminative words are clearly not the most relevant nor the most irrelevant features for topic classification. They are slightly more topic-relevant features than topic-irrelevant but not by a significant margin. The bottom 5000 features for gender discrimination are more strongly topic-irrelevant words.

These results show that gender linguistic differences are not merely isolated in a set of words that would function as markers of gender identity but are rather closely intertwined with semantics. We attempted to improve topic classification by training gender-dependent topic models but we did not observe any gains.

4.5 Can gender lexical differences be exploited to improve automatic speech recognition?

Are the observed gender linguistic differences valuable from an engineering perspective as well? In other words, can a natural language processing task benefit from modeling these differences? In this subsection, we train gender-dependent language models and compare their perplexities with standard baselines. An advantage of using gender information for automatic speech recognition is that it can be robustly detected using acoustic features. In Tables 8 and 9 the perplexities of different gender-dependent language models are shown. The SRILM toolkit (Stolcke, 2002) was used for training the language models using Kneser-Ney smoothing (Kneser and Ney, 1987). The perplexities reported include the end-of-turn as a separate token. 2300 conversation sides are used for training each one of {FF,FM,MF,MM} models of Table 8, while 7670 conversation sides are used for training each one of {F,M} models of Table 9. In both tables, the same 1678 sides are used for testing.

Table 8: Perplexity of gender-dependent bigram language models. Four gender categories are used. Each column has the perplexities for a given test set, each row for a train set.

	FF	**FM**	**MF**	**MM**
FF	**85.3**	91.1	96.5	99.9
FM	85.7	**90.0**	94.5	97.5
MF	87.8	91.4	**93.3**	95.4
MM	89.9	93.1	94.1	**95.2**
ALL	**82.1**	**86.3**	**89.8**	**91.7**

In Tables 8 and 9 we observe that we get lower perplexities in matched than mismatched conditions in training and testing. This is another way to show that different data do exhibit different properties. However, the best results are obtained by pooling all the data and training a single language model. Therefore, despite the fact there are different modes, the benefit of more training data outweighs the benefit of gender-dependent models. Interpolating ALL with F and ALL with M resulted in insignificant improvements (81.6 for F and 89.3 for M).

Table 9: Perplexity of gender-dependent bigram language models. Two gender categories are used. Each column has the perplexities for a given test set, each row for a train set.

	F	**M**
F	**82.8**	94.2
M	86.0	**90.6**
ALL	**81.8**	**89.5**

5 Conclusions

We have presented evidence of linguistic differences between genders using a large corpus of telephone conversations. We have approached the issue from a purely computational perspective and have shown that differences are profound enough that we can classify the transcript of a conversation side according to the gender of the speaker with accuracy close to 93%. Our computational tools have allowed us to quantitatively show that the gender of one speaker influences the linguistic patterns of the other speaker. Specifically, classifying same-gender conversations can be done with almost perfect accuracy, while evidence of some convergence of male and female linguistic patterns in cross-gender conversations was observed. An analysis of the features revealed that the most characteristic features for males are swear words while for females are family-relation words. Leveraging these differences in simple gender-dependent language models is not a win, but this does not imply that more sophisticated language model training methods cannot help. For example, instead of conditioning every word in the vocabulary on gender we can choose to do so only for the top-N, determined by KL or IG. The probability estimates for the rest of the words will be tied for both genders. Future work will examine empirical differences in other features such as dialog acts or turntaking.

References

C. Cieri, D. Miller, and K. Walker. 2004. The Fisher corpus: a resource for the next generations of speech-to-text. In *4th International Conference on Language Resources and Evaluation, LREC*, pages 69–71.

J. Coates, editor. 1997. *Language and Gender: A Reader*. Blackwell Publishers.

G. Doddington. 2001. Speaker recognition based on idiolectal differences between speakers. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech 2001)*, pages 2251–2254.

P. Eckert and S. McConnell-Ginet, editors. 2003. *Language and Gender*. Cambridge University Press.

G. Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Machine Learning Research*, 3:1289–1305.

S. Kiesling. in press. Dude. *American Speech*.

R. Kneser and H. Ney. 1987. Improved backing-off for m-gram language modeling. In *Proc. Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 181–184.

M. Koppel, S. Argamon, and A.R. Shimoni. 2002. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4):401–412.

A. McCallum. 1996. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow.

S. Singh. 2001. A pilot study on gender differences in conversational speech on lexical richness measures. *Literary and Linguistic Computing*, 16(3):251–264.

E. Stamatatos, N. Fakotakis, and G. Kokkinakis. 2000. Automatic text categorization in terms of genre and author. *Computational Linguistics*, 26:471–495.

A. Stolcke. 2002. An extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing (ICSLP)*, pages 901–904.

Position Specific Posterior Lattices for Indexing Speech

Ciprian Chelba and **Alex Acero**
Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
{chelba, alexac}@microsoft.com

Abstract

The paper presents the Position Specific Posterior Lattice, a novel representation of automatic speech recognition lattices that naturally lends itself to efficient indexing of position information and subsequent relevance ranking of spoken documents using proximity.

In experiments performed on a collection of lecture recordings — MIT iCampus data — the spoken document ranking accuracy was improved by 20% relative over the commonly used baseline of indexing the 1-best output from an automatic speech recognizer. The Mean Average Precision (MAP) increased from 0.53 when using 1-best output to 0.62 when using the new lattice representation. The reference used for evaluation is the output of a standard retrieval engine working on the manual transcription of the speech collection.

Albeit lossy, the PSPL lattice is also much more compact than the ASR 3-gram lattice from which it is computed — which translates in reduced inverted index size as well — at virtually no degradation in word-error-rate performance. Since new paths are introduced in the lattice, the ORACLE accuracy increases over the original ASR lattice.

1 Introduction

Ever increasing computing power and connectivity bandwidth together with falling storage costs result in an overwhelming amount of data of various types being produced, exchanged, and stored. Consequently, search has emerged as a key application as more and more data is being saved (Church, 2003). Text search in particular is the most active area, with applications that range from web and intranet search to searching for private information residing on one's hard-drive.

Speech search has not received much attention due to the fact that large collections of untranscribed spoken material have not been available, mostly due to storage constraints. As storage is becoming cheaper, the availability and usefulness of large collections of spoken documents is limited strictly by the lack of adequate technology to exploit them.

Manually transcribing speech is expensive and sometimes outright impossible due to privacy concerns. This leads us to exploring an automatic approach to searching and navigating spoken document collections.

Our current work aims at extending the standard keyword search paradigm from text documents to spoken documents. In order to deal with limitations of current automatic speech recognition (ASR) technology we propose an approach that uses recognition lattices — which are considerably more accurate than the ASR 1-best output.

A novel contribution is the use of a representation of ASR lattices which retains only position information for each word. The Position Specific Posterior

Lattice (PSPL) is a lossy but compact representation of a speech recognition lattice that lends itself to the standard inverted indexing done in text search — which retains the position as well as other contextual information for each hit.

Since our aim is to bridge the gap between text and speech -grade search technology, we take as our reference the output of a text retrieval engine that runs on the manual transcription.

The rest of the paper is structured as follows: in the next section we review previous work in the area, followed by Section 3 which presents a brief overview of state-of-the-art text search technology. We then introduce the PSPL representation in Section 4 and explain its use for indexing and searching speech in the next section. Experiments evaluating ASR accuracy on iCampus, highlighting empirical aspects of PSPL lattices as well as search accuracy results are reported in Section 6. We conclude by outlining future work.

2 Previous Work

The main research effort aiming at spoken document retrieval (SDR) was centered around the SDR-TREC evaluations (Garofolo et al., 2000), although there is a large body of work in this area prior to the SDR-TREC evaluations, as well as more recent work outside this community. Most notable are the contributions of (Brown et al., 1996) and (James, 1995).

One problem encountered in work published prior or outside the SDR-TREC community is that it doesn't always evaluate performance from a document retrieval point of view — using a metric like Mean Average Precision (MAP) or similar, see `trec_eval` (NIST, www) — but rather uses word-spotting measures, which are more technology- rather than user- centric. *We believe that ultimately it is the document retrieval performance that matters and the word-spotting accuracy is just an indicator for how a SDR system might be improved.*

The TREC-SDR 8/9 evaluations — (Garofolo et al., 2000) Section 6 — focused on using Broadcast News speech from various sources: CNN, ABC, PRI, Voice of America. About 550 hrs of speech were segmented manually into 21,574 stories each comprising about 250 words on the average. The approximate manual transcriptions — closed captioning for video — used for SDR system comparison with text-only retrieval performance had fairly high WER: 14.5% video and 7.5% radio broadcasts. ASR systems tuned to the Broadcast News domain were evaluated on detailed manual transcriptions and were able to achieve 15-20% WER, not far from the accuracy of the approximate manual transcriptions. In order to evaluate the accuracy of retrieval systems, search queries —"topics" — along with binary relevance judgments were compiled by human assessors.

SDR systems indexed the ASR 1-best output and their retrieval performance — measured in terms of MAP — was found to be flat with respect to ASR WER variations in the range of 15%-30%. Simply having a common task and an evaluation-driven collaborative research effort represents a huge gain for the community. There are shortcomings however to the SDR-TREC framework.

It is well known that ASR systems are very brittle to mismatched training/test conditions and it is unrealistic to expect error rates in the range 10-15% when decoding speech mismatched with respect to the training data. It is thus very important to consider ASR operating points which have higher WER.

Also, the out-of-vocabulary (OOV) rate was very low, below 1%. Since the "topics"/queries were long and stated in plain English rather than using the keyword search paradigm, the query-side OOV (Q-OOV) was very low as well, an unrealistic situation in practice. (Woodland et al., 2000) evaluates the effect of Q-OOV rate on retrieval performance by reducing the ASR vocabulary size such that the Q-OOV rate comes closer to 15%, a much more realistic figure since search keywords are typically rare words. They show severe degradation in MAP performance — 50% relative, from 44 to 22.

The most common approach to dealing with OOV query words is to represent both the query and the spoken document using sub-word units — typically phones or phone n-grams — and then match sequences of such units. In his thesis, (Ng, 2000) shows the feasibility of sub-word SDR and advocates for tighter integration between ASR and IR technology. Similar conclusions are drawn by the excellent work in (Siegler, 1999).

As pointed out in (Logan et al., 2002), word level indexing and querying is still more accurate, were it not for the OOV problem. The authors argue in favor of a combination of word and sub-word level indexing. Another problem pointed out by the paper is the abundance of word-spotting false-positives in the sub-word retrieval case, somewhat masked by the MAP measure.

Similar approaches are taken by (Seide and Yu, 2004). One interesting feature of this work is a two-pass system whereby an approximate match is carried out at the document level after which the costly detailed phonetic match is carried out on only 15% of the documents in the collection.

More recently, (Saraclar and Sproat, 2004) shows improvement in word-spotting accuracy by using lattices instead of 1-best. An inverted index from symbols — word or phone — to links allows to evaluate adjacency of query words but more general proximity information is harder to obtain — see Section 4. Although no formal comparison has been carried out, we believe our approach should yield a more compact index.

Before discussing our architectural design decisions it is probably useful to give a brief presentation of a state-of-the-art text document retrieval engine that is using the keyword search paradigm.

3 Text Document Retrieval

Probably the most widespread text retrieval model is the TF-IDF vector model (Baeza-Yates and Ribeiro-Neto, 1999). For a given query $\mathcal{Q} = q_1 \ldots q_i \ldots q_Q$ and document D_j one calculates a similarity measure by accumulating the TF-IDF score $w_{i,j}$ for each query term q_i, possibly weighted by a document specific weight:

$$\begin{aligned} S(D_j, \mathcal{Q}) &= \sum_{i=1}^{Q} w_{i,j} \\ w_{i,j} &= f_{i,j} \cdot idf_i \end{aligned}$$

where $f_{i,j}$ is the normalized frequency of word q_i in document D_j and the inverse document frequency for query term q_i is $idf_i = \log \frac{N}{n_i}$ where N is the total number of documents in the collection and n_i is the number of documents containing q_i.

The main criticism to the TF-IDF relevance score is the fact that the query terms are assumed to be independent. *Proximity information* is not taken into account at all, e.g. whether the words LANGUAGE and MODELING occur next to each other or not in a document is not used for relevance scoring.

Another issue is that query terms may be encountered in different *contexts* in a given document: title, abstract, author name, font size, etc. For hypertext document collections even more context information is available: anchor text, as well as other mark-up tags designating various parts of a given document being just a few examples. The TF-IDF ranking scheme completely discards such information although it is clearly important in practice.

3.1 Early Google Approach

Aside from the use of PageRank for relevance ranking, (Brin and Page, 1998) also uses both *proximity* and *context* information heavily when assigning a relevance score to a given document — see Section 4.5.1 of (Brin and Page, 1998) for details.

For each given query term q_i one retrieves the list of *hits* corresponding to q_i in document D. Hits can be of various types depending on the *context* in which the hit occurred: title, anchor text, etc. Each type of hit has its own *type-weight* and the type-weights are indexed by type.

For a single word query, their ranking algorithm takes the inner-product between the type-weight vector and a vector consisting of count-weights (tapered counts such that the effect of large counts is discounted) and combines the resulting score with PageRank in a final relevance score.

For multiple word queries, terms co-occurring in a given document are considered as forming different *proximity-types* based on their proximity, from adjacent to “not even close”. Each proximity type comes with a proximity-weight and the relevance score includes the contribution of proximity information by taking the inner product over all types, including the proximity ones.

3.2 Inverted Index

Of essence to fast retrieval on static document collections of medium to large size is the use of an *inverted index*. The inverted index stores a list of hits for each word in a given vocabulary. The hits are grouped by document. For each document, the list of hits for a given query term must include position — needed to evaluate counts of proximity types —

as well as all the context information needed to calculate the relevance score of a given document using the scheme outlined previously. For details, the reader is referred to (Brin and Page, 1998), Section 4.

4 Position Specific Posterior Lattices

As highlighted in the previous section, position information is crucial for being able to evaluate proximity information when assigning a relevance score to a given document.

In the spoken document case however, we are faced with a dilemma. On one hand, using 1-best ASR output as the transcription to be indexed is suboptimal due to the high WER, which is likely to lead to low recall — query terms that were in fact spoken are wrongly recognized and thus not retrieved. On the other hand, ASR lattices do have much better WER — in our case the 1-best WER was 55% whereas the lattice WER was 30% — but the position information is not readily available: it is easy to evaluate whether two words are adjacent but questions about the distance in number of links between the occurrences of two query words in the lattice are very hard to answer.

The position information needed for recording a given word hit is not readily available in ASR lattices — for details on the format of typical ASR lattices and the information stored in such lattices the reader is referred to (Young et al., 2002). To simplify the discussion let's consider that a traditional text-document hit for given word consists of just `(document id, position)`.

The occurrence of a given word in a lattice obtained from a given spoken document is uncertain and so is the position at which the word occurs in the document.

The ASR lattices do contain the information needed to evaluate proximity information, since on a given path through the lattice we can easily assign a position index to each link/word in the normal way. Each path occurs with a given posterior probability, easily computable from the lattice, so in principle one could index *soft-hits* which specify

```
(document id, position,
              posterior probability)
```

for each word in the lattice. Since it is likely that

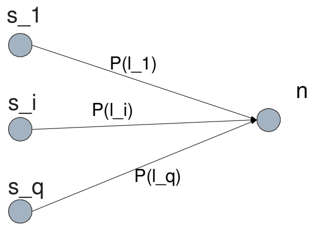


Figure 1: State Transitions

more than one path contains the same word in the same position, one would need to sum over all possible paths in a lattice that contain a given word at a given position.

A simple dynamic programming algorithm which is a variation on the standard forward-backward algorithm can be employed for performing this computation. The computation for the backward pass stays unchanged, whereas during the forward pass one needs to split the forward probability arriving at a given node n, α_n, according to the length l — measured in number of links along the partial path that contain a word; null (ϵ) links are not counted when calculating path length — of the partial paths that start at the start node of the lattice and end at node n:

$$\alpha_n[l] \doteq \sum_{\pi:end(\pi)=n,length(\pi)=l} P(\pi)$$

The backward probability β_n has the standard definition (Rabiner, 1989).

To formalize the calculation of the position-specific forward-backward pass, the initialization, and one elementary forward step in the forward pass are carried out using Eq. (1), respectively — see Figure 1 for notation:

$$\begin{aligned}\alpha_n[l+1] &= \sum_{i=1}^{q} \alpha_{s_i}[l+\delta(l_i,\epsilon)] \cdot P(l_i) \\ \alpha_{start}[l] &= \begin{cases} 1.0, l = 0 \\ 0.0, l \neq 0 \end{cases} \end{aligned} \quad (1)$$

The "probability" $P(l_i)$ of a given link l_i is stored as a $\log$-probability and commonly evaluated in ASR using:

$$\log P(l_i) = FLATw \cdot [1/LMw \cdot \log P_{AM}(l_i) + \log P_{LM}(word(l_i)) - 1/LMw \cdot logP_{IP}] \quad (2)$$

where $\log P_{AM}(l_i)$ is the acoustic model score, $\log P_{LM}(word(l_i))$ is the language model score, $LMw > 0$ is the language model weight, $logP_{IP} > 0$ is the "insertion penalty" and $FLATw$ is a flattening weight. In N-gram lattices where $N \geq 2$, all links ending at a given node n must contain the same word $word(n)$, so the posterior probability of a given word w occurring at a given position l can be easily calculated using:

$$P(w, l|LAT) = \\ \sum_{n \ s.t. \ \alpha_n[l] \cdot \beta_n > 0} \frac{\alpha_n[l] \cdot \beta_n}{\beta_{start}} \cdot \delta(w, word(n))$$

The Position Specific Posterior Lattice (PSPL) is a representation of the $P(w, l|LAT)$ distribution: for each position bin l store the words w along with their posterior probability $P(w, l|LAT)$.

5 Spoken Document Indexing and Search Using PSPL

Spoken documents rarely contain only speech. Often they have a title, author and creation date. There might also be a text abstract associated with the speech, video or even slides in some standard format. The idea of saving *context information* when indexing HTML documents and web pages can thus be readily used for indexing spoken documents, although the context information is of a different nature.

As for the actual *speech content* of a spoken document, the previous section showed how ASR technology and PSPL lattices can be used to automatically convert it to a format that allows the indexing of *soft hits* — a *soft index* stores posterior probability along with the position information for term occurrences in a given document.

5.1 Speech Content Indexing Using PSPL

Speech content can be very long. In our case the speech content of a typical spoken document was approximately 1 hr long; it is customary to segment a given speech file in shorter segments.

A spoken document thus consists of an ordered list of segments. For each segment we generate a corresponding PSPL lattice. Each document and each segment in a given collection are mapped to an integer value using a *collection descriptor file* which lists all documents and segments. Each *soft hit* in our index will store the PSPL position and posterior probability.

5.2 Speech Content Relevance Ranking Using PSPL Representation

Consider a given query $\mathcal{Q} = q_1 \ldots q_i \ldots q_Q$ and a spoken document D represented as a PSPL. Our ranking scheme follows the description in Section 3.1.

The words in the document D clearly belong to the ASR vocabulary $\mathcal{V}$ whereas the words in the query may be out-of-vocabulary (OOV). As argued in Section 2, the query-OOV rate is an important factor in evaluating the impact of having a finite ASR vocabulary on the retrieval accuracy. We assume that the words in the query are all contained in $\mathcal{V}$; OOV words are mapped to UNK and cannot be matched in any document D.

For all query terms, a 1-gram score is calculated by summing the PSPL posterior probability across all segments s and positions k. This is equivalent to calculating the expected count of a given query term q_i according to the PSPL probability distribution $P(w_k(s)|D)$ for each segment s of document D. The results are aggregated in a common value $S_{1-gram}(D, \mathcal{Q})$:

$$S(D, q_i) = \log \left[1 + \sum_s \sum_k P(w_k(s) = q_i|D) \right]$$

$$S_{1-gram}(D, \mathcal{Q}) = \sum_{i=1}^{Q} S(D, q_i) \qquad (3)$$

Similar to (Brin and Page, 1998), the logarithmic tapering off is used for discounting the effect of large counts in a given document.

Our current ranking scheme takes into account proximity in the form of matching N-grams present in the query. Similar to the 1-gram case, we calculate an expected tapered-count for each N-gram $q_i \ldots q_{i+N-1}$ in the query and then aggregate the results in a common value $S_{N-gram}(D, \mathcal{Q})$ for each order N:

$$S(D, q_i \ldots q_{i+N-1}) = \qquad (4) \\ \log \left[1 + \sum_s \sum_k \prod_{l=0}^{N-1} P(w_{k+l}(s) = q_{i+l}|D) \right]$$

$$S_{N-gram}(D, \mathcal{Q}) = \sum_{i=1}^{Q-N+1} S(D, q_i \ldots q_{i+N-1})$$

The different proximity types, one for each N-gram order allowed by the query length, are combined by taking the inner product with a vector of weights.

$$S(D, \mathcal{Q}) = \sum_{N=1}^{Q} w_N \cdot S_{N-gram}(D, \mathcal{Q}) \quad (5)$$

Only documents containing all the terms in the query are returned. In the current implementation the weights increase linearly with the N-gram order. Clearly, better weight assignments must exist, and as the hit types are enriched beyond using just N-grams, the weights will have to be determined using machine learning techniques.

It is worth noting that the transcription for any given segment can also be represented as a PSPL with exactly one word per position bin. It is easy to see that in this case the relevance scores calculated according to Eq. (3-4) are the ones specified by 3.1.

6 Experiments

We have carried all our experiments on the iCampus corpus prepared by MIT CSAIL. The main advantages of the corpus are: realistic speech recording conditions — all lectures are recorded using a lapel microphone — and the availability of accurate manual transcriptions — which enables the evaluation of a SDR system against its text counterpart.

6.1 iCampus Corpus

The iCampus corpus (Glass et al., 2004) consists of about 169 hours of lecture materials: 20 Introduction to Computer Programming Lectures (21.7 hours), 35 Linear Algebra Lectures (27.7 hours), 35 Electro-magnetic Physics Lectures (29.1 hours), 79 Assorted MIT World seminars covering a wide variety of topics (89.9 hours). Each lecture comes with a word-level manual transcription that segments the text into semantic units that could be thought of as sentences; word-level time-alignments between the transcription and the speech are also provided. The speech style is in between planned and spontaneous. The speech is recorded at a sampling rate of 16kHz (wide-band) using a lapel microphone.

The speech was segmented at the sentence level based on the time alignments; each lecture is considered to be a spoken document consisting of a set of one-sentence long segments determined this way — see Section 5.1. The final collection consists of 169 documents, 66,102 segments and an average document length of 391 segments.

We have then used a standard large vocabulary ASR system for generating 3-gram ASR lattices and PSPL lattices. The 3-gram language model used for decoding is trained on a large amount of text data, primarily newswire text. The vocabulary of the ASR system consisted of 110kwds, selected based on frequency in the training data. The acoustic model is trained on a variety of wide-band speech and it is a standard clustered tri-phone, 3-states-per-phone model. Neither model has been tuned in any way to the iCampus scenario.

On the first lecture `L01` of the Introduction to Computer Programming Lectures the WER of the ASR system was 44.7%; the OOV rate was 3.3%. For the entire set of lectures in the Introduction to Computer Programming Lectures, the WER was 54.8%, with a maximum value of 74% and a minimum value of 44%.

6.2 PSPL lattices

We have then proceeded to generate 3-gram lattices and PSPL lattices using the above ASR system. Table 1 compares the accuracy/size of the 3-gram lattices and the resulting PSPL lattices for the first lecture `L01`. As it can be seen the PSPL representation is much more compact than the original 3-gram lattices at a very small loss in accuracy: the 1-best path through the PSPL lattice is only 0.3% absolute worse than the one through the original 3-gram lattice. As expected, the main reduction comes from the drastically smaller node density — 7 times smaller, measured in nodes per word in the reference transcription. Since the PSPL representation

Lattice Type	3-gram	PSPL
Size on disk	11.3MB	3.2MB
Link density	16.3	14.6
Node density	7.4	1.1
1-best WER	44.7%	45%
ORACLE WER	26.4%	21.7%

Table 1: Comparison between 3-gram and PSPL lattices for lecture L01 (iCampus corpus): node and link density, 1-best and ORACLE WER, size on disk

introduces new paths compared to the original 3-gram lattice, the ORACLE WER path — least errorful path in the lattice — is also about 20% relative better than in the original 3-gram lattice — 5% absolute. Also to be noted is the much better WER in both PSPL/3-gram lattices versus 1-best.

6.3 Spoken Document Retrieval

Our aim is to narrow the gap between speech and text document retrieval. We have thus taken as our reference the output of a standard retrieval engine working according to one of the TF-IDF flavors, see Section 3. The engine indexes the manual transcription using an unlimited vocabulary. All retrieval results presented in this section have used the standard `trec_eval` package used by the TREC evaluations.

The PSPL lattices for each segment in the spoken document collection were indexed as explained in 5.1. In addition, we generated the PSPL representation of the manual transcript and of the 1-best ASR output and indexed those as well. This allows us to compare our retrieval results against the results obtained using the reference engine when working on the same text document collection.

6.3.1 Query Collection and Retrieval Setup

The missing ingredient for performing retrieval experiments are the queries. We have asked a few colleagues to issue queries against a demo shell using the index built from the manual transcription. The only information[1] provided to them was the same as the summary description in Section 6.1.

We have collected 116 queries in this manner. The query out-of-vocabulary rate (Q-OOV) was 5.2% and the average query length was 1.97 words. Since our approach so far does not index sub-word units, we cannot deal with OOV query words. We have thus removed the queries which contained OOV words — resulting in a set of 96 queries — which clearly biases the evaluation. On the other hand, the results on both the 1-best and the lattice indexes are equally favored by this.

[1]Arguably, more motivated users that are also more familiar with the document collection would provide a better query collection framework

6.3.2 Retrieval Experiments

We have carried out retrieval experiments in the above setup. Indexes have been built from:

- `trans`: manual transcription filtered through ASR vocabulary
- `1-best`: ASR 1-best output
- `lat`: PSPL lattices.

No tuning of retrieval weights, see Eq. (5), or link scoring weights, see Eq. (2) has been performed. Table 2 presents the results. As a sanity check, the retrieval results on transcription — `trans` — match almost perfectly the reference. The small difference comes from stemming rules that the baseline engine is using for query enhancement which are not replicated in our retrieval engine. The results on lattices (`lat`) improve significantly on (`1-best`) — 20% relative improvement in mean average precision (MAP).

	`trans`	`1-best`	`lat`
# docs retrieved	1411	3206	4971
# relevant docs	1416	1416	1416
# rel retrieved	1411	1088	1301
MAP	0.99	0.53	0.62
R-precision	0.99	0.53	0.58

Table 2: Retrieval performance on indexes built from transcript, ASR 1-best and PSPL lattices, respectively

6.3.3 Why Would This Work?

A legitimate question at this point is: *why would anyone expect this to work when the 1-best ASR accuracy is so poor?*

In favor of our approach, the ASR lattice WER is much lower than the 1-best WER, and PSPL have even lower WER than the ASR lattices. As reported in Table 1, the PSPL WER for `L01` was 22% whereas the 1-best WER was 45%. Consider matching a 2-gram in the PSPL —the average query length is indeed 2 wds so this is a representative situation. A simple calculation reveals that it is twice — $(1 - 0.22)^2/(1 - 0.45)^2 = 2$ — more likely to find a query match in the PSPL than in the 1-best — if the query 2-gram was indeed spoken at that position. According to this heuristic argument one could expect a dramatic increase in Recall. Another aspect is that people enter *typical N-grams* as queries. The contents of adjacent PSPL bins are fairly random in nature so if a typical 2-gram is found in the PSPL, chances are it was actually spoken. This translates in little degradation in Precision.

7 Conclusions and Future work

We have developed a new representation for ASR lattices — the Position Specific Posterior Lattice (**PSPL**) — that lends itself naturally to indexing speech content and integrating state-of-the-art IR techniques that make use of *proximity and context* information. In addition, the PSPL representation is also much more compact at no loss in WER — both 1-best and ORACLE.

The retrieval results obtained by indexing the PSPL and performing adequate relevance ranking are 20% better than when using the ASR 1-best output, although still far from the performance achieved on text data.

The experiments presented in this paper are truly a first step. We plan to gather a much larger number of queries. The binary relevance judgments — a given document is deemed either relevant or irrelevant to a given query in the reference "ranking" — assumed by the standard `trec_eval` tool are also a serious shortcoming; a distance measure between *rankings* of documents needs to be used. Finally, using a baseline engine that in fact makes use of proximity and context information is a priority if such information is to be used in our algorithms.

8 Acknowledgments

We would like to thank Jim Glass and T J Hazen at MIT for providing the iCampus data. We would also like to thank Frank Seide for offering valuable suggestions and our colleagues for providing queries.

References

Ricardo Baeza-Yates and Berthier Ribeiro-Neto, 1999. *Modern Information Retrieval*, chapter 2, pages 27–30. Addison Wesley, New York.

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.

M. G. Brown, J. T. Foote, G. J. F. Jones, K. Spärck Jones, and S. J. Young. 1996. Open-vocabulary speech indexing for voice and video mail retrieval. In *Proc. ACM Multimedia 96*, pages 307–316, Boston, November.

Kenneth Ward Church. 2003. Speech and language processing: Where have we been and where are we going? In *Proceedings of Eurospeech*, Geneva, Switzerland.

J. Garofolo, G. Auzanne, and E. Voorhees. 2000. The TREC spoken document retrieval track: A success story. In *Proceedings of the Recherche d'Informations Assiste par Ordinateur: ContentBased Multimedia Information Access Conference*, April.

James Glass, T. J. Hazen, Lee Hetherington, and Chao Wang. 2004. Analysis and processing of lecture audio data: Preliminary investigations. In *HLT-NAACL 2004 Workshop: Interdisciplinary Approaches to Speech Indexing and Retrieval*, pages 9–12, Boston, Massachusetts, May.

David Anthony James. 1995. *The Application of Classical Information Retrieval Techniques to Spoken Documents*. Ph.D. thesis, University of Cambridge, Downing College.

B. Logan, P. Moreno, and O. Deshmukh. 2002. Word and sub-word indexing approaches for reducing the effects of OOV queries on spoken audio. In *Proc. HLT*.

Kenney Ng. 2000. *Subword-Based Approaches for Spoken Document Retrieval*. Ph.D. thesis, Massachusetts Institute of Technology.

NIST. www. The TREC evaluation package. In *www-nlpir.nist.gov/projects/trecvid/trecvid.tools/trec_eval*.

L. R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings IEEE*, volume 77(2), pages 257–285.

Murat Saraclar and Richard Sproat. 2004. Lattice-based search for spoken utterance retrieval. In *HLT-NAACL 2004*, pages 129–136, Boston, Massachusetts, May.

F. Seide and P. Yu. 2004. Vocabulary-independent search in spontaneous speech. In *Proceedings of ICASSP*, Montreal, Canada.

Matthew A. Siegler. 1999. *Integration of Continuous Speech Recognition and Information Retrieval for Mutually Optimal Performance*. Ph.D. thesis, Carnegie Mellon University.

P. C. Woodland, S. E. Johnson, P. Jourlin, and K. Spärck Jones. 2000. Effects of out of vocabulary words in spoken document retrieval. In *Proceedings of SIGIR*, pages 372–374, Athens, Greece.

Steve Young, Gunnar Evermann, Thomas Hain, Dan Kershaw, Gareth Moore, Julian Odell, Dan Povey Dave Ollason, Valtcho Valtchev, and Phil Woodland. 2002. *The HTK Book*. Cambridge University Engineering Department, Cambridge, England, December.

Using Conditional Random Fields For Sentence Boundary Detection In Speech

Yang Liu
ICSI, Berkeley
yangl@icsi.berkeley.edu

Andreas Stolcke **Elizabeth Shriberg**
SRI and ICSI
stolcke,ees@speech.sri.com

Mary Harper
Purdue University
harper@ecn.purdue.edu

Abstract

Sentence boundary detection in speech is important for enriching speech recognition output, making it easier for humans to read and downstream modules to process. In previous work, we have developed hidden Markov model (HMM) and maximum entropy (Maxent) classifiers that integrate textual and prosodic knowledge sources for detecting sentence boundaries. In this paper, we evaluate the use of a conditional random field (CRF) for this task and relate results with this model to our prior work. We evaluate across two corpora (conversational telephone speech and broadcast news speech) on both human transcriptions and speech recognition output. In general, our CRF model yields a lower error rate than the HMM and Maxent models on the NIST sentence boundary detection task in speech, although it is interesting to note that the best results are achieved by three-way voting among the classifiers. This probably occurs because each model has different strengths and weaknesses for modeling the knowledge sources.

1 Introduction

Standard speech recognizers output an unstructured stream of words, in which the important structural features such as sentence boundaries are missing. Sentence segmentation information is crucial and assumed in most of the further processing steps that one would want to apply to such output: tagging and parsing, information extraction, summarization, among others.

1.1 Sentence Segmentation Using HMM

Most prior work on sentence segmentation (Shriberg et al., 2000; Gotoh and Renals, 2000; Christensen et al., 2001; Kim and Woodland, 2001; NIST-RT03F, 2003) have used an HMM approach, in which the word/tag sequences are modeled by N-gram language models (LMs) (Stolcke and Shriberg, 1996). Additional features (mostly related to speech prosody) are modeled as observation likelihoods attached to the N-gram states of the HMM (Shriberg et al., 2000). Figure 1 shows the graphical model representation of the variables involved in the HMM for this task. Note that the words appear in both the states[1] and the observations, such that the word stream constrains the possible hidden states to matching words; the ambiguity in the task stems entirely from the choice of events. This architecture differs from the one typically used for sequence tagging (e.g., part-of-speech tagging), in which the "hidden" states represent only the events or tags. Empirical investigations have shown that omitting words in the states significantly degrades system performance for sentence boundary detection (Liu, 2004). The observation probabilities in the HMM, implemented using a decision tree classifier, capture the probabilities of generating the prosodic features

[1] In this sense, the states are only partially "hidden".

$P(F_i|E_i, W_i)$.[2] An N-gram LM is used to calculate the transition probabilities:

$$
\begin{aligned}
P(W_iE_i|W_1E_1 \dots W_{i-1}E_{i-1}) = \\
P(W_i|W_1E_1 \dots W_{i-1}E_{i-1}) \times \\
P(E_i|W_1E_1 \dots W_{i-1}E_{i-1}E_i)
\end{aligned}
$$

In the HMM, the forward-backward algorithm is used to determine the event with the highest posterior probability for each interword boundary:

$$\hat{E}_i = \arg\max_{E_i} P(E_i|W, F) \qquad (1)$$

The HMM is a generative modeling approach since it describes a stochastic process with hidden variables (sentence boundary) that produces the observable data. This HMM approach has two main drawbacks. First, standard training methods maximize the joint probability of observed and hidden events, as opposed to the posterior probability of the correct hidden variable assignment given the observations, which would be a criterion more closely related to classification performance. Second, the N-gram LM underlying the HMM transition model makes it difficult to use features that are highly correlated (such as words and POS labels) without greatly increasing the number of model parameters, which in turn would make robust estimation difficult. More details about using textual information in the HMM system are provided in Section 3.

1.2 Sentence Segmentation Using Maxent

A maximum entropy (Maxent) posterior classification method has been evaluated in an attempt to overcome some of the shortcomings of the HMM approach (Liu et al., 2004; Huang and Zweig, 2002). For a boundary position i, the Maxent model takes the exponential form:

$$P(E_i|T_i, F_i) = \frac{1}{Z_\lambda(T_i, F_i)} e^{\sum_k \lambda_k g_k(E_i, T_i, F_i)} \qquad (2)$$

where $Z_\lambda(T_i, F_i)$ is a normalization term and T_i represents textual information. The indicator functions $g_k(E_i, T_i, F_i)$ correspond to features defined over events, words, and prosody. The parameters in

[2] In the prosody model implementation, we ignore the word identity in the conditions, only using the timing or word alignment information.

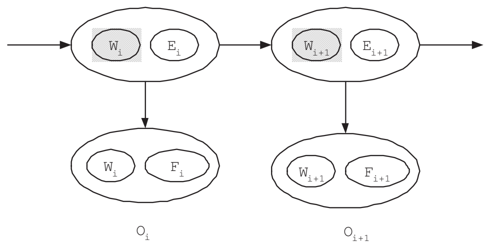


Figure 1: A graphical model of HMM for the sentence boundary detection problem. Only one word+event pair is depicted in each state, but in a model based on N-grams, the previous $N - 1$ tokens would condition the transition to the next state. O are observations consisting of words W and prosodic features F, and E are sentence boundary events.

Maxent are chosen to maximize the conditional likelihood $\prod_i P(E_i|T_i, F_i)$ over the training data, better matching the classification accuracy metric. The Maxent framework provides a more principled way to combine the largely correlated textual features, as confirmed by the results of (Liu et al., 2004); however, it does not model the state sequence.

A simple combination of the results from the Maxent and HMM was found to improve upon the performance of either model alone (Liu et al., 2004) because of the complementary strengths and weaknesses of the two models. An HMM is a generative model, yet it is able to model the sequence via the forward-backward algorithm. Maxent is a discriminative model; however, it attempts to make decisions locally, without using sequential information.

A conditional random field (CRF) model (Lafferty et al., 2001) combines the benefits of the HMM and Maxent approaches. Hence, in this paper we will evaluate the performance of the CRF model and relate the results to those using the HMM and Maxent approaches on the sentence boundary detection task. The rest of the paper is organized as follows. Section 2 describes the CRF model and discusses how it differs from the HMM and Maxent models. Section 3 describes the data and features used in the models to be compared. Section 4 summarizes the experimental results for the sentence boundary detection task. Conclusions and future work appear in Section 5.

2 CRF Model Description

A CRF is a random field that is globally conditioned on an observation sequence O. CRFs have been successfully used for a variety of text processing tasks (Lafferty et al., 2001; Sha and Pereira, 2003; McCallum and Li, 2003), but they have not been widely applied to a speech-related task with both acoustic and textual knowledge sources. The top graph in Figure 2 is a general CRF model. The states of the model correspond to event labels E. The observations O are composed of the textual features, as well as the prosodic features. The most likely event sequence $\hat{E}$ for the given input sequence (observations) O is

$$\hat{E} = \arg\max_{E} \frac{e^{\sum_k \lambda_k G_k(E,O)}}{Z_\lambda(O)} \tag{3}$$

where the functions G are potential functions over the events and the observations, and Z_λ is the normalization term:

$$Z_\lambda(O) = \sum_{E} e^{\sum_k \lambda_k G_k(E,O)} \tag{4}$$

Even though a CRF itself has no restriction on the potential functions $G_k(E,O)$, to simplify the model (considering computational cost and the limited training set size), we use a first-order CRF in this investigation, as at the bottom of Figure 2. In this model, an observation O_i (consisting of textual features T_i and prosodic features F_i) is associated with a state E_i.

The model is trained to maximize the conditional log-likelihood of a given training set. Similar to the Maxent model, the conditional likelihood is closely related to the individual event posteriors used for classification, enabling this type of model to explicitly optimize discrimination of correct from incorrect labels. The most likely sequence is found using the Viterbi algorithm.[3]

A CRF differs from an HMM with respect to its training objective function (joint versus conditional likelihood) and its handling of dependent word features. Traditional HMM training does not maximize the posterior probabilities of the correct labels; whereas, the CRF directly estimates posterior boundary label probabilities $P(E|O)$. The underlying N-gram sequence model of an HMM does not cope well with multiple representations (features) of the word sequence (e.g., words, POS), especially when the training set is small; however, the CRF model supports simultaneous correlated features, and therefore gives greater freedom for incorporating a variety of knowledge sources. A CRF differs from the Maxent method with respect to its ability to model sequence information. The primary advantage of the CRF over the Maxent approach is that the model is optimized globally over the entire sequence; whereas, the Maxent model makes a local decision, as shown in Equation (2), without utilizing any state dependency information.

We use the Mallet package (McCallum, 2002) to implement the CRF model. To avoid overfitting, we employ a Gaussian prior with a zero mean on the parameters (Chen and Rosenfeld, 1999), similar to what is used for training Maxent models (Liu et al., 2004).

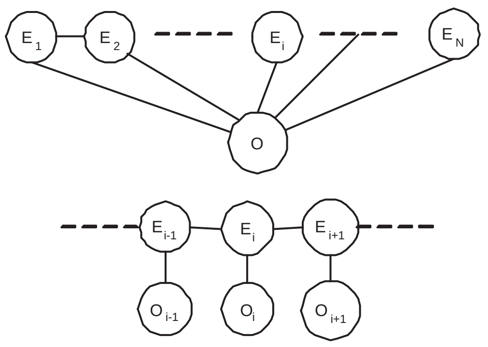


Figure 2: Graphical representations of a general CRF and the first-order CRF used for the sentence boundary detection problem. E represent the state tags (i.e., sentence boundary or not). O are observations consisting of words W or derived textual features T and prosodic features F.

3 Experimental Setup

3.1 Data and Task Description

The sentence-like units in speech are different from those in written text. In conversational speech, these units can be well-formed sentences, phrases, or even a single word. These units are called SUs in the DARPA EARS program. SU boundaries, as

[3]The forward-backward algorithm would most likely be better here, but it is not implemented in the software we used (McCallum, 2002).

well as other structural metadata events, were annotated by LDC according to an annotation guideline (Strassel, 2003). Both the transcription and the recorded speech were used by the annotators when labeling the boundaries.

The SU detection task is conducted on two corpora: Broadcast News (BN) and Conversational Telephone Speech (CTS). BN and CTS differ in genre and speaking style. The average length of SUs is longer in BN than in CTS, that is, 12.35 words (standard deviation 8.42) in BN compared to 7.37 words (standard deviation 8.72) in CTS. This difference is reflected in the frequency of SU boundaries: about 14% of interword boundaries are SUs in CTS compared to roughly 8% in BN. Training and test data for the SU detection task are those used in the NIST Rich Transcription 2003 Fall evaluation. We use both the development set and the evaluation set as the test set in this paper in order to obtain more meaningful results. For CTS, there are about 40 hours of conversational data (around 480K words) from the Switchboard corpus for training and 6 hours (72 conversations) for testing. The BN data has about 20 hours of Broadcast News shows (about 178K words) in the training set and 3 hours (6 shows) in the test set. Note that the SU-annotated training data is only a subset of the data used for the speech recognition task because more effort is required to annotate the boundaries.

For testing, the system determines the locations of sentence boundaries given the word sequence W and the speech. The SU detection task is evaluated on both the reference human transcriptions (REF) and speech recognition outputs (STT). Evaluation across transcription types allows us to obtain the performance for the best-case scenario when the transcriptions are correct; thus factoring out the confounding effect of speech recognition errors on the SU detection task. We use the speech recognition output obtained from the SRI recognizer (Stolcke et al., 2003).

System performance is evaluated using the official NIST evaluation tools.[4] System output is scored by first finding a minimum edit distance alignment between the hypothesized word string and the reference transcriptions, and then comparing the aligned event labels. The SU error rate is defined as the total number of deleted or inserted SU boundary events, divided by the number of true SU boundaries. In addition to this **NIST SU error metric**, we use the total number of interword boundaries as the denominator, and thus obtain results for the **per-boundary-based metric**.

[4]See http://www.nist.gov/speech/tests/rt/rt2003/fall/ for more details about scoring.

3.2 Feature Extraction and Modeling

To obtain a good-quality estimation of the conditional probability of the event tag given the observations $P(E_i|O_i)$, the observations should be based on features that are discriminative of the two events (SU versus not). As in (Liu et al., 2004), we utilize both textual and prosodic information.

We extract prosodic features that capture duration, pitch, and energy patterns associated with the word boundaries (Shriberg et al., 2000). For all the modeling methods, we adopt a modular approach to model the prosodic features, that is, a decision tree classifier is used to model them. During testing, the decision tree prosody model estimates posterior probabilities of the events given the associated prosodic features for a word boundary. The posterior probability estimates are then used in various modeling approaches in different ways as described later.

Since words and sentence boundaries are mutually constraining, the word identities themselves (from automatic recognition or human transcriptions) constitute a primary knowledge source for sentence segmentation. We also make use of various automatic taggers that map the word sequence to other representations. Tagged versions of the word stream are provided to support various generalizations of the words and to smooth out possibly undertrained word-based probability estimates. These tags include part-of-speech tags, syntactic chunk tags, and automatically induced word classes. In addition, we use extra text corpora, which were not annotated according to the guideline used for the training and test data (Strassel, 2003). For BN, we use the training corpus for the LM for speech recognition. For CTS, we use the Penn Treebank Switchboard data. There is punctuation information in both, which we use to approximate SUs as defined in the annotation guideline (Strassel, 2003).

As explained in Section 1, the prosody model and

Table 1: Knowledge sources and their representations in different modeling approaches: HMM, Maxent, and CRF.

	HMM	Maxent	CRF
	generative model	conditional approach	
Sequence information	yes	no	yes
LDC data set (words or tags)	LM	N-grams as indicator functions	
Probability from prosody model	real-valued	cumulatively binned	
Additional text corpus	N-gram LM	binned posteriors	
Speaker turn change	in prosodic features	a separate feature, in addition to being in the prosodic feature set	
Compound feature	no	POS tags and decisions from prosody model	

the N-gram LM can be integrated in an HMM. When various textual information is used, jointly modeling words and tags may be an effective way to model the richer feature set; however, a joint model requires more parameters. Since the training set for the SU detection task in the EARS program is quite limited, we use a loosely coupled approach:

- Linearly combine three LMs: the word-based LM from the LDC training data, the automatic-class-based LMs, and the word-based LM trained from the additional corpus.
- These interpolated LMs are then combined with the prosody model via the HMM. The posterior probabilities of events at each boundary are obtained from this step, denoted as $P_{HMM}(E_i|W,C,F)$.
- Apply the POS-based LM alone to the POS sequence (obtained by running the POS tagger on the word sequence W) and generate the posterior probabilities for each word boundary $P_{posLM}(E_i|POS)$, which are then combined from the posteriors from the previous step, i.e., $P_{final}(E_i|T,F) = P_{HMM}(E_i|W,C,F) + P_{posLM}(E_i|P)$.

The features used for the CRF are the same as those used for the Maxent model devised for the SU detection task (Liu et al., 2004), briefly listed below.

- N-grams of words or various tags (POS tags, automatically induced classes). Different Ns and different position information are used (N varies from one through four).
- The cumulative binned posterior probabilities from the decision tree prosody model.
- The N-gram LM trained from the extra corpus is used to estimate posterior event probabilities for the LDC-annotated training and test sets, and these posteriors are then thresholded to yield binary features.
- Other features: speaker or turn change, and compound features of POS tags and decisions from the prosody model.

Table 1 summarizes the features and their representations used in the three modeling approaches. The same knowledge sources are used in these approaches, but with different representations. The goal of this paper is to evaluate the ability of these three modeling approaches to combine prosodic and textual knowledge sources, not in a rigidly parallel fashion, but by exploiting the inherent capabilities of each approach. We attempt to compare the models in as parallel a fashion as possible; however, it should be noted that the two discriminative methods better model the textual sources and the HMM better models prosody given its representation in this study.

4 Experimental Results and Discussion

SU detection results using the CRF, HMM, and Maxent approaches individually, on the reference transcriptions or speech recognition output, are shown in Tables 2 and 3 for CTS and BN data, respectively. We present results when different knowledge sources are used: word N-gram only, word N-gram and prosodic information, and using all the

Table 2: Conversational telephone speech SU detection results reported using the NIST SU error rate (%) and the boundary-based error rate (% in parentheses) using the HMM, Maxent, and CRF individually and in combination. Note that the 'all features' condition uses all the knowledge sources described in Section 3.2. 'Vote' is the result of the majority vote over the three modeling approaches, each of which uses all the features. The baseline error rate when assuming there is no SU boundary at each word boundary is 100% for the NIST SU error rate and 15.7% for the boundary-based metric.

Conversational Telephone Speech				
		HMM	Maxent	CRF
REF	word N-gram	42.02 (6.56)	43.70 (6.82)	37.71 (5.88)
	word N-gram + prosody	33.72 (5.26)	35.09 (5.47)	30.88 (4.82)
	all features	31.51 (4.92)	30.66 (4.78)	29.47 (4.60)
		Vote: 29.30 (4.57)		
STT	word N-gram	53.25 (8.31)	53.92 (8.41)	50.20 (7.83)
	word N-gram + prosody	44.93 (7.01)	45.50 (7.10)	43.12 (6.73)
	all features	43.05 (6.72)	43.02 (6.71)	42.00 (6.55)
		Vote: 41.88 (6.53)		

features described in Section 3.2. The word N-grams are from the LDC training data and the extra text corpora. 'All the features' means adding textual information based on tags, and the 'other features' in the Maxent and CRF models as well. The detection error rate is reported using the NIST SU error rate, as well as the per-boundary-based classification error rate (in parentheses in the table) in order to factor out the effect of the different SU priors. Also shown in the tables are the majority vote results over the three modeling approaches when all the features are used.

4.1 CTS Results

For CTS, we find from Table 2 that the CRF is superior to both the HMM and the Maxent model across all conditions (the differences are significant at $p < 0.05$). When using only the word N-gram information, the gain of the CRF is the greatest, with the differences among the models diminishing as more features are added. This may be due to the impact of the sparse data problem on the CRF or simply due to the fact that differences among modeling approaches are less when features become stronger, that is, the good features compensate for the weaknesses in models. Notice that with fewer knowledge sources (e.g., using only word N-gram and prosodic information), the CRF is able to achieve performance similar to or even better than other methods using all the knowledges sources. This may be useful when feature extraction is computationally expensive.

We observe from Table 2 that there is a large increase in error rate when evaluating on speech recognition output. This happens in part because word information is inaccurate in the recognition output, thus impacting the effectiveness of the LMs and lexical features. The prosody model is also affected, since the alignment of incorrect words to the speech is imperfect, thereby degrading prosodic feature extraction. However, the prosody model is more robust to recognition errors than textual knowledge, because of its lesser dependence on word identity. The results show that the CRF suffers most from the recognition errors. By focusing on the results when only word N-gram information is used, we can see the effect of word errors on the models. The SU detection error rate increases more in the STT condition for the CRF model than for the other models, suggesting that the discriminative CRF model suffers more from the mismatch between the training (using the reference transcription) and the test condition (features obtained from the errorful words).

We also notice from the CTS results that when only word N-gram information is used (with or without combining with prosodic information), the HMM is superior to the Maxent; only when various additional textual features are included in the feature set does Maxent show its strength compared to

Table 3: Broadcast news SU detection results reported using the NIST SU error rate (%) and the boundary-based error rate (% in parentheses) using the HMM, Maxent, and CRF individually and in combination. The baseline error rate is 100% for the NIST SU error rate and 7.2% for the boundary-based metric.

Broadcast News				
		HMM	Maxent	CRF
REF	word N-gram	80.44 (5.83)	81.30 (5.89)	74.99 (5.43)
	word N-gram + prosody	59.81 (4.33)	59.69 (4.33)	54.92 (3.98)
	all features	48.72 (3.53)	48.61 (3.52)	47.92 (3.47)
		Vote: 46.28 (3.35)		
STT	word N-gram	84.71 (6.14)	86.13 (6.24)	80.50 (5.83)
	word N-gram + prosody	64.58 (4.68)	63.16 (4.58)	59.52 (4.31)
	all features	55.37 (4.01)	56.51 (4.10)	55.37 (4.01)
		Vote: 54.29 (3.93)		

the HMM, highlighting the benefit of Maxent's handling of the textual features.

The combined result (using majority vote) of the three approaches in Table 2 is superior to any model alone (the improvement is not significant though). Previously, it was found that the Maxent and HMM posteriors combine well because the two approaches have different error patterns (Liu et al., 2004). For example, Maxent yields fewer insertion errors than HMM because of its reliance on different knowledge sources. The toolkit we use for the implementation of the CRF does not generate a posterior probability for a sequence; therefore, we do not combine the system output via posterior probability interpolation, which is expected to yield better performance.

4.2 BN Results

Table 3 shows the SU detection results for BN. Similar to the patterns found for the CTS data, the CRF consistently outperforms the HMM and Maxent, except on the STT condition when all the features are used. The CRF yields relatively less gain over the other approaches on BN than on CTS. One possible reason for this difference is that there is more training data for the CTS task, and both the CRF and Maxent approaches require a relatively larger training set than the HMM. Overall the degradation on the STT condition for BN is smaller than on CTS. This can be easily explained by the difference in word error rates, 22.9% on CTS and 12.1% on BN. Finally, the vote among the three approaches outperforms any model on both the REF and STT conditions, and the gain from voting is larger for BN than CTS.

Comparing Table 2 and Table 3, we find that the NIST SU error rate on BN is generally higher than on CTS. This is partly because the NIST error rate is measured as the percentage of errors per reference SU, and the number of SUs in CTS is much larger than for BN, giving a large denominator and a relatively lower error rate for the same number of boundary detection errors. Another reason is that the training set is smaller for BN than for CTS. Finally, the two genres differ significantly: CTS has the advantage of the frequent backchannels and first person pronouns that provide good cues for SU detection. When the boundary-based classification metric is used (results in parentheses), the SU error rate is lower on BN than on CTS; however, it should also be noted that the baseline error rate (i.e., the priors of the SUs) is lower on BN than CTS.

5 Conclusion and Future Work

Finding sentence boundaries in speech transcriptions is important for improving readability and aiding downstream language processing modules. In this paper, prosodic and textual knowledge sources are integrated for detecting sentence boundaries in speech. We have shown that a discriminatively trained CRF model is a competitive approach for the sentence boundary detection task. The CRF combines the advantages of being discriminatively trained and able to model the entire sequence, and so it outperforms the HMM and Maxent approaches consistently across various testing conditions. The CRF takes longer to train than the HMM and Maxent models, especially when the number of features becomes large; the HMM requires the least training time of all approaches. We also find that as more features are used, the differences among the modeling approaches decrease. We have explored different approaches to modeling various knowledge sources in an attempt to achieve good performance for sentence boundary detection. Note that we have not fully optimized each modeling approach. For example, for the HMM, using discriminative training methods is likely to improve system performance, but possibly at a cost of reducing the accuracy of the combined system.

In future work, we will examine the effect of Viterbi decoding versus forward-backward decoding for the CRF approach, since the latter better matches the classification accuracy metric. To improve SU detection results on the STT condition, we plan to investigate approaches that model recognition uncertainty in order to mitigate the effect of word errors. Another future direction is to investigate how to effectively incorporate prosodic features more directly in the Maxent or CRF framework, rather than using a separate prosody model and then binning the resulting posterior probabilities.

Important ongoing work includes investigating the impact of SU detection on downstream language processing modules, such as parsing. For these applications, generating probabilistic SU decisions is crucial since that information can be more effectively used by subsequent modules.

6 Acknowledgments

The authors thank the anonymous reviewers for their valuable comments, and Andrew McCallum and Aron Culotta at the University of Massachusetts and Fernando Pereira at the University of Pennsylvania for their assistance with their CRF toolkit. This work has been supported by DARPA under contract MDA972-02-C-0038, NSF-STIMULATE under IRI-9619921, NSF KDI BCS-9980054, and ARDA under contract MDA904-03-C-1788. Distribution is unlimited. Any opinions expressed in this paper are those of the authors and do not reflect the funding agencies. Part of the work was carried out while the last author was on leave from Purdue University and at NSF.

References

S. Chen and R. Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University.

H. Christensen, Y. Gotoh, and S. Renal. 2001. Punctuation annotation using statistical prosody models. In *ISCA Workshop on Prosody in Speech Recognition and Understanding*.

Y. Gotoh and S. Renals. 2000. Sentence boundary detection in broadcast speech transcripts. In *Proceedings of ISCA Workshop: Automatic Speech Recognition: Challenges for the New Millennium ASR-2000*, pages 228–235.

J. Huang and G. Zweig. 2002. Maximum entropy model for punctuation annotation from speech. In *Proceedings of the International Conference on Spoken Language Processing*, pages 917–920.

J. Kim and P. C. Woodland. 2001. The use of prosody in a combined system for punctuation generation and speech recognition. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 2757–2760.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random field: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289.

Y. Liu, A. Stolcke, E. Shriberg, and M. Harper. 2004. Comparing and combining generative and posterior probability models: Some advances in sentence boundary detection in speech. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Y. Liu. 2004. *Structural Event Detection for Rich Transcription of Speech*. Ph.D. thesis, Purdue University.

A. McCallum and W. Li. 2003. Early results for named entity recognition with conditional random fields. In *Proceedings of the Conference on Computational Natural Language Learning*.

A. McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

NIST-RT03F. 2003. RT-03F workshop agenda and presentations. http://www.nist.gov/speech/tests/rt/rt2003/fall/presentations/, November.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting*.

E. Shriberg, A. Stolcke, D. Hakkani-Tur, and G. Tur. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, pages 127–154.

A. Stolcke and E. Shriberg. 1996. Automatic linguistic segmentation of conversational speech. In *Proceedings of the International Conference on Spoken Language Processing*, pages 1005–1008.

A. Stolcke, H. Franco, R. Gadde, M. Graciarena, K. Precoda, A. Venkataraman, D. Vergyri, W. Wang, and J. Zheng. 2003. Speech-to-text research at SRI-ICSI-UW. http://www.nist.gov/speech/tests/rt/rt2003/spring/presentations/index.htm.

S. Strassel, 2003. *Simple Metadata Annotation Specification V5.0*. Linguistic Data Consortium.

Log-linear Models for Word Alignment

Yang Liu , Qun Liu and **Shouxun Lin**
Institute of Computing Technology
Chinese Academy of Sciences
No. 6 Kexueyuan South Road, Haidian District
P. O. Box 2704, Beijing, 100080, China
{yliu, liuqun, sxlin}@ict.ac.cn

Abstract

We present a framework for word alignment based on log-linear models. All knowledge sources are treated as feature functions, which depend on the source langauge sentence, the target language sentence and possible additional variables. Log-linear models allow statistical alignment models to be easily extended by incorporating syntactic information. In this paper, we use IBM Model 3 alignment probabilities, POS correspondence, and bilingual dictionary coverage as features. Our experiments show that log-linear models significantly outperform IBM translation models.

1 Introduction

Word alignment, which can be defined as an object for indicating the corresponding words in a parallel text, was first introduced as an intermediate result of statistical translation models (Brown et al., 1993). In statistical machine translation, word alignment plays a crucial role as word-aligned corpora have been found to be an excellent source of translation-related knowledge.

Various methods have been proposed for finding word alignments between parallel texts. There are generally two categories of alignment approaches: *statistical approaches* and *heuristic approaches*. Statistical approaches, which depend on a set of unknown parameters that are learned from training data, try to describe the relationship between a bilingual sentence pair (Brown et al., 1993; Vogel and Ney, 1996). Heuristic approaches obtain word alignments by using various similarity functions between the types of the two languages (Smadja et al., 1996; Ker and Chang, 1997; Melamed, 2000). The central distinction between statistical and heuristic approaches is that statistical approaches are based on well-founded probabilistic models while heuristic ones are not. Studies reveal that statistical alignment models outperform the simple Dice coefficient (Och and Ney, 2003).

Finding word alignments between parallel texts, however, is still far from a trivial work due to the diversity of natural languages. For example, the alignment of words within idiomatic expressions, free translations, and missing content or function words is problematic. When two languages widely differ in word order, finding word alignments is especially hard. Therefore, it is necessary to incorporate all useful linguistic information to alleviate these problems.

Tiedemann (2003) introduced a word alignment approach based on combination of association clues. Clues combination is done by disjunction of single clues, which are defined as probabilities of associations. The crucial assumption of clue combination that clues are independent of each other, however, is not always true. Och and Ney (2003) proposed Model 6, a log-linear combination of IBM translation models and HMM model. Although Model 6 yields better results than naive IBM models, it fails to include dependencies other than IBM models and HMM model. Cherry and Lin (2003) developed a statistical model to find word alignments, which allow easy integration of context-specific features.

Log-linear models, which are very suitable to incorporate additional dependencies, have been successfully applied to statistical machine translation (Och and Ney, 2002). In this paper, we present a framework for word alignment based on log-linear models, allowing statistical models to be easily extended by incorporating additional syntactic dependencies. We use IBM Model 3 alignment probabilities, POS correspondence, and bilingual dictionary coverage as features. Our experiments show that log-linear models significantly outperform IBM translation models.

We begin by describing log-linear models for word alignment. The design of feature functions is discussed then. Next, we present the training method and the search algorithm for log-linear models. We will follow with our experimental results and conclusion and close with a discussion of possible future directions.

2 Log-linear Models

Formally, we use following definition for alignment. Given a source ('English') sentence $\mathbf{e} = e_1^I = e_1, \ldots, e_i, \ldots, e_I$ and a target language ('French') sentence $\mathbf{f} = f_1^J = f_1, \ldots, f_j, \ldots, f_J$. We define a link $l = (i, j)$ to exist if e_i and f_j are translation (or part of a translation) of one another. We define the null link $l = (i, 0)$ to exist if e_i does not correspond to a translation for any French word in $\mathbf{f}$. The null link $l = (0, j)$ is defined similarly. An alignment $\mathbf{a}$ is defined as a subset of the Cartesian product of the word positions:

$$\mathbf{a} \subseteq \{(i, j) : i = 0, \ldots, I; j = 0, \ldots, J\} \quad (1)$$

We define the alignment problem as finding the alignment $\mathbf{a}$ that maximizes $Pr(\mathbf{a} \mid \mathbf{e}, \mathbf{f})$ given $\mathbf{e}$ and $\mathbf{f}$.

We directly model the probability $Pr(\mathbf{a} \mid \mathbf{e}, \mathbf{f})$. An especially well-founded framework is maximum entropy (Berger et al., 1996). In this framework, we have a set of M feature functions $h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})$, $m = 1, \ldots, M$. For each feature function, there exists a model parameter λ_m, $m = 1, \ldots, M$. The direct alignment probability is given by:

$$Pr(\mathbf{a}|\mathbf{e}, \mathbf{f}) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})]}{\sum_{\mathbf{a}'} \exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a}', \mathbf{e}, \mathbf{f})]} \quad (2)$$

This approach has been suggested by (Papineni et al., 1997) for a natural language understanding task and successfully applied to statistical machine translation by (Och and Ney, 2002).

We obtain the following decision rule:

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmax}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(\mathbf{a}, \mathbf{e}, \mathbf{f}) \right\} \quad (3)$$

Typically, the source language sentence $\mathbf{e}$ and the target sentence $\mathbf{f}$ are the fundamental knowledge sources for the task of finding word alignments. Linguistic data, which can be used to identify associations between lexical items are often ignored by traditional word alignment approaches. Linguistic tools such as part-of-speech taggers, parsers, named-entity recognizers have become more and more robust and available for many languages by now. It is important to make use of linguistic information to improve alignment strategies. Treated as feature functions, syntactic dependencies can be easily incorporated into log-linear models.

In order to incorporate a new dependency which contains extra information other than the bilingual sentence pair, we modify Eq.2 by adding a new variable $\mathbf{v}$:

$$Pr(\mathbf{a}|\mathbf{e}, \mathbf{f}, \mathbf{v}) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a}, \mathbf{e}, \mathbf{f}, \mathbf{v})]}{\sum_{\mathbf{a}'} \exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a}', \mathbf{e}, \mathbf{f}, \mathbf{v})]} \quad (4)$$

Accordingly, we get a new decision rule:

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmax}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(\mathbf{a}, \mathbf{e}, \mathbf{f}, \mathbf{v}) \right\} \quad (5)$$

Note that our log-linear models are different from Model 6 proposed by Och and Ney (2003), which defines the alignment problem as finding the alignment $\mathbf{a}$ that maximizes $Pr(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$ given $\mathbf{e}$.

3 Feature Functions

In this paper, we use IBM translation Model 3 as the base feature of our log-linear models. In addition, we also make use of syntactic information such as part-of-speech tags and bilingual dictionaries.

3.1 IBM Translation Models

Brown et al. (1993) proposed a series of statistical models of the translation process. IBM translation models try to model the translation probability $Pr(f_1^J|e_1^I)$, which describes the relationship between a source language sentence e_1^I and a target language sentence f_1^J. In statistical alignment models $Pr(f_1^J, a_1^J|e_1^I)$, a 'hidden' alignment $\mathbf{a} = a_1^J$ is introduced, which describes a mapping from a target position j to a source position $i = a_j$. The relationship between the translation model and the alignment model is given by:

$$Pr(f_1^J|e_1^I) = \sum_{a_1^J} Pr(f_1^J, a_1^J|e_1^I) \tag{6}$$

Although IBM models are considered more coherent than heuristic models, they have two drawbacks. First, IBM models are restricted in a way such that each target word f_j is assigned to exactly one source word e_{a_j}. A more general way is to model alignment as an arbitrary relation between source and target language positions. Second, IBM models are typically language-independent and may fail to tackle problems occurred due to specific languages.

In this paper, we use Model 3 as our base feature function, which is given by [1]:

$$\begin{aligned} h(\mathbf{a}, \mathbf{e}, \mathbf{f}) &= Pr(f_1^J, a_1^J|e_1^I) \\ = &\binom{m-\phi_0}{\phi_0} {p_0}^{m-2\phi_0} {p_1}^{\phi_0} \prod_{i=1}^{l} \phi_i! n(\phi_i|e_i) \times \\ &\prod_{j=1}^{m} t(f_j|e_{a_j}) d(j|a_j, l, m) \end{aligned} \tag{7}$$

We distinguish between two translation directions to use Model 3 as feature functions: treating English as source language and French as target language or vice versa.

3.2 POS Tags Transition Model

The first linguistic information we adopt other than the source language sentence $\mathbf{e}$ and the target language sentence $\mathbf{f}$ is part-of-speech tags. The use of POS information for improving statistical alignment quality of the HMM-based model is described in (Toutanova et al., 2002). They introduce additional lexicon probability for POS tags in both languages.

In IBM models as well as HMM models, when one needs the model to take new information into account, one must create an extended model which can base its parameters on the previous model. In log-linear models, however, new information can be easily incorporated.

We use a POS Tags Transition Model as a feature function. This feature learns POS Tags transition probabilities from held-out data (via simple counting) and then applies the learned distributions to the ranking of various word alignments. We define $\mathbf{eT} = eT_1^I = eT_1, \ldots, eT_i, \ldots, eT_I$ and $\mathbf{fT} = fT_1^J = fT_1, \ldots, fT_j, \ldots, fT_J$ as POS tag sequences of the sentence pair $\mathbf{e}$ and $\mathbf{f}$. POS Tags Transition Model is formally described as:

$$Pr(\mathbf{fT}|\mathbf{a}, \mathbf{eT}) = \prod_{a} t(fT_{a(j)}|eT_{a(i)}) \tag{8}$$

where a is an element of $\mathbf{a}$, $a(i)$ is the corresponding source position of a and $a(j)$ is the target position.

Hence, the feature function is:

$$h(\mathbf{a}, \mathbf{e}, \mathbf{f}, \mathbf{eT}, \mathbf{fT}) = \prod_{a} t(fT_{a(j)}|eT_{a(i)}) \tag{9}$$

We still distinguish between two translation directions to use POS tags Transition Model as feature functions: treating English as source language and French as target language or vice versa.

3.3 Bilingual Dictionary

A conventional bilingual dictionary can be considered an additional knowledge source. We could use a feature that counts how many entries of a conventional lexicon co-occur in a given alignment between the source sentence and the target sentence. Therefore, the weight for the provided conventional dictionary can be learned. The intuition is that the conventional dictionary is expected to be more reliable than the automatically trained lexicon and therefore should get a larger weight.

We define a bilingual dictionary as a set of entries: $\mathbf{D} = \{(e, f, conf)\}$. e is a source language word, f is a target langauge word, and $conf$ is a positive real-valued number (usually, $conf = 1.0$) assigned

[1] If there is a target word which is assigned to more than one source words, $h(\mathbf{a}, \mathbf{e}, \mathbf{f}) = 0$.

by lexicographers to evaluate the validity of the entry. Therefore, the feature function using a bilingual dictionary is:

$$h(\mathbf{a}, \mathbf{e}, \mathbf{f}, \mathbf{D}) = \sum_{a} occur(e_{a(i)}, f_{a(j)}, D) \quad (10)$$

where

$$occur(e, f, D) = \begin{cases} conf & \text{if } (e, f) \text{ occurs in } D \\ 0 & \text{else} \end{cases} \quad (11)$$

4 Training

We use the GIS (Generalized Iterative Scaling) algorithm (Darroch and Ratcliff, 1972) to train the model parameters λ_1^M of the log-linear models according to Eq. 4. By applying suitable transformations, the GIS algorithm is able to handle any type of real-valued features. In practice, We use YASMET [2] written by Franz J. Och for performing training.

The renormalization needed in Eq. 4 requires a sum over a large number of possible alignments. If $\mathbf{e}$ has length l and $\mathbf{f}$ has length m, there are possible 2^{lm} alignments between $\mathbf{e}$ and $\mathbf{f}$ (Brown et al., 1993). It is unrealistic to enumerate all possible alignments when lm is very large. Hence, we approximate this sum by sampling the space of all possible alignments by a large set of highly probable alignments. The set of considered alignments are also called n-best list of alignments.

We train model parameters on a development corpus, which consists of hundreds of manually-aligned bilingual sentence pairs. Using an n-best approximation may result in the problem that the parameters trained with the GIS algorithm yield worse alignments even on the development corpus. This can happen because with the modified model scaling factors the n-best list can change significantly and can include alignments that have not been taken into account in training. To avoid this problem, we iteratively combine n-best lists to train model parameters until the resulting n-best list does not change, as suggested by Och (2002). However, as this training procedure is based on maximum likelihood criterion, there is only a loose relation to the final alignment quality on unseen bilingual texts. In practice, having a series of model parameters when the iteration ends, we select the model parameters that yield best alignments on the development corpus.

[2] Available at http://www.fjoch.com/YASMET.html

After the bilingual sentences in the development corpus are tokenized (or segmented) and POS tagged, they can be used to train POS tags transition probabilities by counting relative frequencies:

$$p(fT|eT) = \frac{N_A(fT, eT)}{N(eT)}$$

Here, $N_A(fT, eT)$ is the frequency that the POS tag fT is aligned to POS tag eT and $N(eT)$ is the frequency of eT in the development corpus.

5 Search

We use a greedy search algorithm to search the alignment with highest probability in the space of all possible alignments. A state in this space is a partial alignment. A transition is defined as the addition of a single link to the current state. Our start state is the empty alignment, where all words in $\mathbf{e}$ and $\mathbf{f}$ are assigned to null. A terminal state is a state in which no more links can be added to increase the probability of the current alignment. Our task is to find the terminal state with the highest probability.

We can compute $gain$, which is a heuristic function, instead of probability for efficiency. A gain is defined as follows:

$$gain(\mathbf{a}, l) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a} \cup l, \mathbf{e}, \mathbf{f})]}{\exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})]} \quad (12)$$

where $l = (i, j)$ is a link added to $\mathbf{a}$.

The greedy search algorithm for general log-linear models is formally described as follows:

Input: $\mathbf{e}, \mathbf{f}, \mathbf{eT}, \mathbf{fT}$, and $\mathbf{D}$

Output: $\mathbf{a}$

```
1. Start with $\mathbf{a} = \phi$.
2. Do for each $l = (i, j)$ and $l \notin \mathbf{a}$:
     Compute $gain(\mathbf{a}, l)$
3. Terminate if $\forall l, gain(\mathbf{a}, l) \leq 1$.
4. Add the link $\hat{l}$ with the maximal $gain(\mathbf{a}, l)$
   to $\mathbf{a}$.
5. Goto 2.
```

The above search algorithm, however, is not efficient for our log-linear models. It is time-consuming for each feature to figure out a probability when adding a new link, especially when the sentences are very long. For our models, $gain(\mathbf{a}, l)$ can be obtained in a more efficient way [3]:

$$gain(\mathbf{a}, l) = \sum_{m=1}^{M} \lambda_m \log\left(\frac{h_m(\mathbf{a} \cup l, \mathbf{e}, \mathbf{f})}{h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})}\right) \quad (13)$$

Note that we restrict that $h(\mathbf{a}, \mathbf{e}, \mathbf{f}) \geq 0$ for all feature functions.

The original terminational condition for greedy search algorithm is:

$$gain(\mathbf{a}, l) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a} \cup l, \mathbf{e}, \mathbf{f})]}{\exp[\sum_{m=1}^{M} \lambda_m h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})]} \leq 1.0$$

That is:

$$\sum_{m=1}^{M} \lambda_m [h_m(\mathbf{a} \cup l, \mathbf{e}, \mathbf{f}) - h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})] \leq 0.0$$

By introducing gain threshold t, we obtain a new terminational condition:

$$\sum_{m=1}^{M} \lambda_m \log\left(\frac{h_m(\mathbf{a} \cup l, \mathbf{e}, \mathbf{f})}{h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})}\right) \leq t$$

where

$$t = \sum_{m=1}^{M} \lambda_m \left\{ \log\left(\frac{h_m(\mathbf{a} \cup l, \mathbf{e}, \mathbf{f})}{h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})}\right) - [h_m(\mathbf{a} \cup l, \mathbf{e}, \mathbf{f}) - h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})] \right\}$$

Note that we restrict $h(\mathbf{a}, \mathbf{e}, \mathbf{f}) \geq 0$ for all feature functions. Gain threshold t is a real-valued number, which can be optimized on the development corpus.

Therefore, we have a new search algorithm:

Input: $\mathbf{e}, \mathbf{f}, \mathbf{eT}$, **fT**, $\mathbf{D}$ and t

Output: $\mathbf{a}$

```
1. Start with $\mathbf{a} = \phi$.
2. Do for each $l = (i, j)$ and $l \notin \mathbf{a}$:
     Compute $gain(\mathbf{a}, l)$
3. Terminate if $\forall l, gain(\mathbf{a}, l) \leq t$.
4. Add the link $\hat{l}$ with the maximal $gain(\mathbf{a}, l)$
   to $\mathbf{a}$.
5. Goto 2.
```

[3]We still call the new heuristic function $gain$ to reduce notational overhead, although the $gain$ in Eq. 13 is not equivalent to the one in Eq. 12.

The gain threshold t depends on the added link l. We remove this dependency for simplicity when using it in search algorithm by treating it as a fixed real-valued number.

6 Experimental Results

We present in this section results of experiments on a parallel corpus of Chinese-English texts. Statistics for the corpus are shown in Table 1. We use a training corpus, which is used to train IBM translation models, a bilingual dictionary, a development corpus, and a test corpus.

		Chinese	English
Train	Sentences	108 925	
	Words	3 784 106	3 862 637
	Vocabulary	49 962	55 698
Dict	Entries	415 753	
	Vocabulary	206 616	203 497
Dev	Sentences	435	
	Words	11 462	14 252
	Ave. SentLen	26.35	32.76
Test	Sentences	500	
	Words	13 891	15 291
	Ave. SentLen	27.78	30.58

Table 1. Statistics of training corpus (Train), bilingual dictionary (Dict), development corpus (Dev), and test corpus (Test).

The Chinese sentences in both the development and test corpus are segmented and POS tagged by ICTCLAS (Zhang et al., 2003). The English sentences are tokenized by a simple tokenizer of ours and POS tagged by a rule-based tagger written by Eric Brill (Brill, 1995). We manually aligned 935 sentences, in which we selected 500 sentences as test corpus. The remaining 435 sentences are used as development corpus to train POS tags transition probabilities and to optimize the model parameters and gain threshold.

Provided with human-annotated word-level alignment, we use precision, recall and AER (Och and

	Size of Training Corpus				
	1K	5K	9K	39K	109K
Model 3 E $\rightarrow$ C	0.4497	0.4081	0.4009	0.3791	0.3745
Model 3 C $\rightarrow$ E	0.4688	0.4261	0.4221	0.3856	0.3469
Intersection	0.4588	0.4106	0.4044	0.3823	0.3687
Union	0.4596	0.4210	0.4157	0.3824	0.3703
Refined Method	0.4154	0.3586	0.3499	0.3153	0.3068
Model 3 E $\rightarrow$ C	0.4490	0.3987	0.3834	0.3639	0.3533
+ Model 3 C $\rightarrow$ E	0.3970	0.3317	0.3217	0.2949	0.2850
+ POS E $\rightarrow$ C	0.3828	0.3182	0.3082	0.2838	0.2739
+ POS C $\rightarrow$ E	0.3795	0.3160	0.3032	0.2821	0.2726
+ Dict	0.3650	0.3092	0.2982	0.2738	0.2685

Table 2. Comparison of AER for results of using IBM Model 3 (GIZA++) and log-linear models.

Ney, 2003) for scoring the viterbi alignments of each model against gold-standard annotated alignments:

$$\text{precision} = \frac{|A \cap P|}{|A|}$$

$$\text{recall} = \frac{|A \cap S|}{|S|}$$

$$\text{AER} = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

where A is the set of word pairs aligned by word alignment systems, S is the set marked in the gold standard as ”sure” and P is the set marked as ”possible” (including the ”sure” pairs). In our Chinese-English corpus, only one type of alignment was marked, meaning that $S = P$.

In the following, we present the results of log-linear models for word alignment. We used GIZA++ package (Och and Ney, 2003) to train IBM translation models. The training scheme is $1^5H^53^5$, which means that Model 1 are trained for five iterations, HMM model for five iterations and finally Model 3 for five iterations. Except for changing the iterations for each model, we use default configuration of GIZA++. After that, we used three types of methods for performing a symmetrization of IBM models: intersection, union, and refined methods (Och and Ney , 2003).

The base feature of our log-linear models, IBM Model 3, takes the parameters generated by GIZA++ as parameters for itself. In other words, our log-linear models share GIZA++ with the same parameters apart from POS transition probability table and bilingual dictionary.

Table 2 compares the results of our log-linear models with IBM Model 3. From row 3 to row 7 are results obtained by IBM Model 3. From row 8 to row 12 are results obtained by log-linear models.

As shown in Table 2, our log-linear models achieve better results than IBM Model 3 in all training corpus sizes. Considering Model 3 E $\rightarrow$ C of GIZA++ and ours alone, greedy search algorithm described in Section 5 yields surprisingly better alignments than hillclimbing algorithm in GIZA++.

Table 3 compares the results of log-linear models with IBM Model 5. The training scheme is $1^5H^53^54^55^5$. Our log-linear models still make use of the parameters generated by GIZA++.

Comparing Table 3 with Table 2, we notice that our log-linear models yield slightly better alignments by employing parameters generated by the training scheme $1^5H^53^54^55^5$ rather than $1^5H^53^5$, which can be attributed to improvement of parameters after further Model 4 and Model 5 training.

For log-linear models, POS information and an additional dictionary are used, which is not the case for GIZA++/IBM models. However, treated as a method for performing symmetrization, log-linear combination alone yields better results than intersection, union, and refined methods.

Figure 1 shows how gain threshold has an effect on precision, recall and AER with fixed model scaling factors.

Figure 2 shows the effect of number of features

	Size of Training Corpus				
	1K	5K	9K	39K	109K
Model 5 E $\rightarrow$ C	0.4384	0.3934	0.3853	0.3573	0.3429
Model 5 C $\rightarrow$ E	0.4564	0.4067	0.3900	0.3423	0.3239
Intersection	0.4432	0.3916	0.3798	0.3466	0.3267
Union	0.4499	0.4051	0.3923	0.3516	0.3375
Refined Method	0.4106	0.3446	0.3262	0.2878	0.2748
Model 3 E $\rightarrow$ C	0.4372	0.3873	0.3724	0.3456	0.3334
+ Model 3 C $\rightarrow$ E	0.3920	0.3269	0.3167	0.2842	0.2727
+ POS E $\rightarrow$ C	0.3807	0.3122	0.3039	0.2732	0.2667
+ POS C $\rightarrow$ E	0.3731	0.3091	0.3017	0.2722	0.2657
+ Dict	0.3612	0.3046	0.2943	0.2658	0.2625

Table 3. Comparison of AER for results of using IBM Model 5 (GIZA++) and log-linear models.

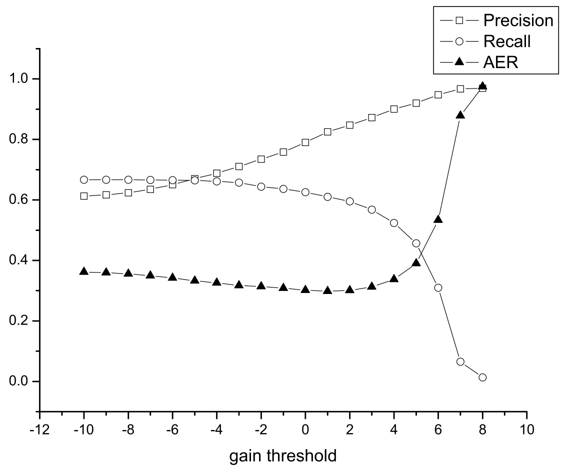


Figure 1. Precision, recall and AER over different gain thresholds with the same model scaling factors.

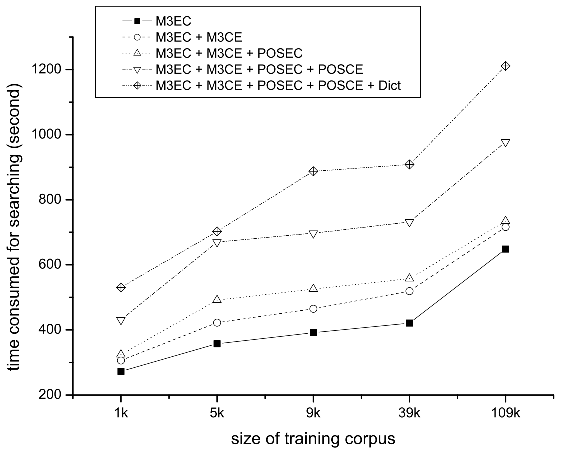


Figure 2. Effect of number of features and size of training corpus on search efficiency.

and size of training corpus on search efficiency for log-linear models.

Table 4 shows the resulting normalized model scaling factors. We see that adding new features also has an effect on the other model scaling factors.

	MEC	+MCE	+PEC	+PCE	+Dict
λ_1	1.000	0.466	0.291	0.202	0.151
λ_2	-	0.534	0.312	0.212	0.167
λ_3	-	-	0.397	0.270	0.257
λ_4	-	-	-	0.316	0.306
λ_5	-	-	-	-	0.119

Table 4. Resulting model scaling factors: λ_1: Model 3 E $\rightarrow$ C (MEC); λ_2: Model 3 C $\rightarrow$ E (MCE); λ_3: POS E $\rightarrow$ C (PEC); λ_4: POS C $\rightarrow$ E (PCE); λ_5: Dict (normalized such that $\sum_{m=1}^{5} \lambda_m = 1$).

7 Conclusion

We have presented a framework for word alignment based on log-linear models between parallel texts. It allows statistical models easily extended by incorporating syntactic information. We take IBM Model 3 as base feature and use syntactic information such as POS tags and bilingual dictionary. Experimental results show that log-linear models for word alignment significantly outperform IBM translation models. However, the search algorithm we proposed is supervised, relying on a hand-aligned bilingual corpus, while the baseline approach of IBM alignments is unsupervised.

Currently, we only employ three types of knowledge sources as feature functions. Syntax-based translation models, such as *tree-to-string* model (Yamada and Knight, 2001) and *tree-to-tree* model (Gildea, 2003), may be very suitable to be added into log-linear models.

It is promising to optimize the model parameters directly with respect to AER as suggested in statistical machine translation (Och, 2003).

Acknowledgement

This work is supported by National High Technology Research and Development Program contract ”Generally Technical Research and Basic Database Establishment of Chinese Platform” (Subject No. 2004AA114010).

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. DellaPietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39-72, March.

Eric Brill. 1995. Transformation-based-error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4), December.

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.

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan.

J. N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470-1480.

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

Sue J. Ker and Jason S. Chang. 1997. A class-based approach to word alignment. *Computational Linguistics*, 23(2):313-343, June.

I. Dan Melamed 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221-249, June.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295-302, Philadelphia, PA, July.

Franz J. Och. 2002. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. Ph.D. thesis, Computer Science Department, RWTH Aachen, Germany, October.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages: 160-167, Sapporo, Japan.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19-51, March.

Kishore A. Papineni, Salim Roukos, and Todd Ward. 1997. Feature-based language understanding. In *European Conf. on Speech Communication and Technology*, pages 1435-1438, Rhodes, Greece, September.

Frank Smadja, Vasileios Hatzivassiloglou, and Kathleen R. McKeown 1996. Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, 22(1):1-38, March.

Jörg Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of the 10th Conference of European Chapter of the ACL (EACL)*, Budapest, Hungary, April.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. 2003. Extensions to HMM-based statistical word alignment models. In *Proceedings of Empirical Methods in Natural Langauge Processing*, Philadelphia, PA.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th Int. Conf. on Computational Linguistics*, pages 836-841, Copenhagen, Denmark, August.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical machine translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages: 523-530, Toulouse, France, July.

Huaping Zhang, Hongkui Yu, Deyi Xiong, and Qun Liu. 2003. HHMM-based Chinese lexical analyzer ICTCLAS. In *Proceedings of the second SigHan Workshop affiliated with 41th ACL*, pages: 184-187, Sapporo, Japan.

Alignment Model Adaptation for Domain-Specific Word Alignment

WU Hua, WANG Haifeng, LIU Zhanyi
Toshiba (China) Research and Development Center
5/F., Tower W2, Oriental Plaza
No.1, East Chang An Ave., Dong Cheng District
Beijing, 100738, China
{wuhua, wanghaifeng, liuzhanyi}@rdc.toshiba.com.cn

Abstract

This paper proposes an alignment adaptation approach to improve domain-specific (in-domain) word alignment. The basic idea of alignment adaptation is to use out-of-domain corpus to improve in-domain word alignment results. In this paper, we first train two statistical word alignment models with the large-scale out-of-domain corpus and the small-scale in-domain corpus respectively, and then interpolate these two models to improve the domain-specific word alignment. Experimental results show that our approach improves domain-specific word alignment in terms of both precision and recall, achieving a relative error rate reduction of 6.56% as compared with the state-of-the-art technologies.

1 Introduction

Word alignment was first proposed as an intermediate result of statistical machine translation (Brown et al., 1993). In recent years, many researchers have employed statistical models (Wu, 1997; Och and Ney, 2003; Cherry and Lin, 2003) or association measures (Smadja et al., 1996; Ahrenberg et al., 1998; Tufis and Barbu, 2002) to build alignment links. In order to achieve satisfactory results, all of these methods require a large-scale bilingual corpus for training. When the large-scale bilingual corpus is not available, some researchers use existing dictionaries to improve word alignment (Ker and Chang, 1997). However, only a few studies (Wu and Wang, 2004) directly address the problem of domain-specific word alignment when neither the large-scale domain-specific bilingual corpus nor the domain-specific translation dictionary is available.

In this paper, we address the problem of word alignment in a specific domain, in which only a small-scale corpus is available. In the domain-specific (in-domain) corpus, there are two kinds of words: general words, which also frequently occur in the out-of-domain corpus, and domain-specific words, which only occur in the specific domain. Thus, we can use the out-of-domain bilingual corpus to improve the alignment for general words and use the in-domain bilingual corpus for domain-specific words. We implement this by using alignment model adaptation.

Although the adaptation technology is widely used for other tasks such as language modeling (Iyer et al., 1997), only a few studies, to the best of our knowledge, directly address word alignment adaptation. Wu and Wang (2004) adapted the alignment results obtained with the out-of-domain corpus to the results obtained with the in-domain corpus. This method first trained two models and two translation dictionaries with the in-domain corpus and the out-of-domain corpus, respectively. Then these two models were applied to the in-domain corpus to get different results. The trained translation dictionaries were used to select alignment links from these different results. Thus, this method performed adaptation through result combination. The experimental results showed a significant error rate reduction as compared with the method directly combining the two corpora as training data.

In this paper, we improve domain-specific word alignment through statistical alignment model adaptation instead of result adaptation. Our method includes the following steps: (1) two word alignment models are trained using a small-scale in-domain bilingual corpus and a large-scale out-of-domain bilingual corpus, respectively. (2) A new alignment model is built by interpolating the two trained models. (3) A translation dictionary is also built by interpolating the two dictionaries that are trained from the two training corpora. (4) The new alignment model and the translation dictionary are employed to improve domain-specific word alignment results. Experimental results show that our approach improves domain-specific word alignment in terms of both precision and recall, achieving a relative error rate reduction of 6.56% as compared with the state-of-the-art technologies.

The remainder of the paper is organized as follows. Section 2 introduces the statistical word alignment model. Section 3 describes our alignment model adaptation method. Section 4 describes the method used to build the translation dictionary. Section 5 describes the model adaptation algorithm. Section 6 presents the evaluation results. The last section concludes our approach.

2 Statistical Word Alignment

According to the IBM models (Brown et al., 1993), the statistical word alignment model can be generally represented as in Equation (1).

$$p(a \mid \boldsymbol{f}, \boldsymbol{e}) = \frac{p(a, \boldsymbol{f} \mid \boldsymbol{e})}{\sum_{a'} p(a', \boldsymbol{f} \mid \boldsymbol{e})} \qquad (1)$$

In this paper, we use a simplified IBM model 4 (Al-Onaizan et al., 1999), which is shown in Equation (2). This simplified version does not take word classes into account as described in (Brown et al., 1993).

$$\begin{aligned} p(a, \boldsymbol{f} \mid \boldsymbol{e}) &= \sum_{(\tau,\pi)} \Pr(\tau, \pi \mid \boldsymbol{e}) \\ &= \binom{m-\phi_0}{\phi_0} p_0^{m-2\phi_0} p_1^{\phi_0} \cdot \\ &\quad \prod_{i=1}^{l} n(\phi_i \mid e_i) \cdot \prod_{j=1}^{m} t(f_j \mid e_{a_j}) \cdot \\ &\quad \Big(\prod_{j=1, a_j \neq 0}^{m} ([j = h(a_j)] \cdot d_1(j - c_{\rho_{a_j}})) + \\ &\quad \prod_{j=1, a_j \neq 0}^{m} ([j \neq h(a_j)] \cdot d_{>1}(j - p(j)))\Big) \end{aligned} \qquad (2)$$

l, m are the lengths of the target sentence and the source sentence respectively.

j is the position index of the source word.

a_j is the position of the target word aligned to the jth source word.

ϕ_i is the fertility of e_i.

p_1 is the fertility probability for e_0, and $p_0 + p_1 = 1$.

$t(f_j|e_{a_j})$ is the word translation probability.

$n(\phi_i \mid e_i)$ is the fertility probability.

$d_1(j - c_{\rho_{a_j}})$ is the distortion probability for the head of each cept[1].

$d_{>1}(j - p(j))$ is the distortion probability for the remaining words of the cept.

$h(i) = \min_k \{k : i = a_k\}$ is the head of cept *i*.

$p(j) = \max_{k<j} \{k : a_j = a_k\}$

ρ_i is the first word before e_i with non-zero fertility. If $|\{i' : \phi_{i'} > 0 \wedge 0 < i' < i\}| > 0$, $\rho_i = \max\{i' : \phi_{i'} > 0 \wedge 0 < i' < i\}$; else $\rho_i = 0$.

$c_i = \frac{\sum_j [a_j = i] \cdot j}{\phi_i}$ is the center of cept *i*.

During the training process, IBM model 3 is first trained, and then the parameters in model 3 are employed to train model 4. During the testing process, the trained model 3 is also used to get an initial alignment result, and then the trained model 4 is employed to improve this alignment result. For convenience, we describe model 3 in Equation (3). The main difference between model 3 and model 4 lies in the calculation of distortion probability.

$$\begin{aligned} p(a, \boldsymbol{f} \mid \boldsymbol{e}) &= \sum_{(\tau,\pi)} \Pr(\tau, \pi \mid \boldsymbol{e}) \\ &= \binom{m-\phi_0}{\phi_0} p_0^{m-2\phi_0} p_1^{\phi_0} \cdot \\ &\quad \prod_{i=1}^{l} n(\phi_i \mid e_i) \cdot \prod_{i=1}^{l} \phi_i! \cdot \\ &\quad \prod_{j=1}^{m} t(f_j \mid e_{a_j}) \cdot \prod_{j:a_j \neq 0}^{m} d(j \mid a_j, l, m) \end{aligned} \qquad (3)$$

[1] A cept is defined as the set of target words connected to a source word (Brown et al., 1993).

However, both model 3 and model 4 do not take the multiword cept into account. Only one-to-one and many-to-one word alignments are considered. Thus, some multi-word units in the domain-specific corpus cannot be correctly aligned. In order to deal with this problem, we perform word alignment in two directions (source to target, and target to source) as described in (Och and Ney, 2000). The GIZA++ toolkit[2] is used to perform statistical word alignment.

We use SG_1 and SG_2 to represent the bi-directional alignment sets, which are shown in Equation (4) and (5). For alignment in both sets, we use *j* for source words and *i* for target words. If a target word in position *i* is connected to source words in positions j_1 and j_2, then $A_i = \{j_1, j_2\}$. We call an element in the alignment set an *alignment link.*

$$SG_1 = \{(A_i, i) \mid A_i = \{j \mid a_j = i, a_j \geq 0\}\} \quad (4)$$

$$SG_2 = \{(j, A_j) \mid A_j = \{i \mid i = a_j,\ a_j \geq 0\}\} \quad (5)$$

3 Word Alignment Model Adaptation

In this paper, we first train two models using the out-of-domain training data and the in-domain training data, and then build a new alignment model through linear interpolation of the two trained models. In other words, we make use of the out-of-domain training data and the in-domain training data by interpolating the trained alignment models. One method to perform model adaptation is to directly interpolate the alignment models as shown in Equation (6).

$$p(a \mid f, e) = \lambda \cdot p_I(a \mid f, e) + (1 - \lambda) \cdot p_O(a \mid f, e) \quad (6)$$

$p_I(a \mid f, e)$ and $p_O(a \mid f, e)$ are the alignment model trained using the in-domain corpus and the out-of-domain corpus, respectively. λ is an interpolation weight. It can be a constant or a function of f and e.

However, in both model 3 and model 4, there are mainly three kinds of parameters: translation probability, fertility probability and distortion probability. These three kinds of parameters have their own interpretation in these two models. In order to obtain fine-grained interpolation models, we separate the alignment model interpolation into three parts: translation probability interpolation, fertility probability interpolation and distortion probability interpolation. For these probabilities, we use different interpolation methods to calculate the interpolation weights. After interpolation, we replace the corresponding parameters in equation (2) and (3) with the interpolated probabilities to get new alignment models.

In the following subsections, we will perform linear interpolation for word alignment in the source to target direction. For the word alignment in the target to source direction, we use the same interpolation method.

3.1 Translation Probability Interpolation

The word translation probability $t(f_j \mid e_{a_j})$ is very important in translation models. The same word may have different distributions in the in-domain corpus and the out-of-domain corpus. Thus, the interpolation weight for the translation probability is taken as a variant. The interpolation model for $t(f_j \mid e_{a_j})$ is described in Equation (7).

$$t(f_j \mid e_{a_j}) = \lambda_t(e_{a_j}) \cdot t_I(f_j \mid e_{a_j}) + (1 - \lambda_t(e_{a_j})) \cdot t_O(f_j \mid e_{a_j}) \quad (7)$$

The interpolation weight $\lambda_t(e_{a_j})$ in (7) is a function of e_{a_j}. It is calculated as shown in Equation (8).

$$\lambda_t(e_{a_j}) = \left(\frac{p_I(e_{a_j})}{p_I(e_{a_j}) + p_O(e_{a_j})} \right)^{\alpha} \quad (8)$$

$p_I(e_{a_j})$ and $p_O(e_{a_j})$ are the relative frequencies of e_{a_j} in the in-domain corpus and in the out-of-domain corpus, respectively. α is an adaptation coefficient, such that $\alpha \geq 0$.

Equation (8) indicates that if a word occurs more frequently in a specific domain than in the general domain, it can usually be considered as a domain-specific word (Peñas et al., 2001). For example, if $p_I(e_{a_j})$ is much larger than $p_O(e_{a_j})$, the word e_{a_j} is a domain-specific word and the interpolation weight approaches to 1. In this case, we trust more on the translation probability obtained from the in-domain corpus than that obtained from the out-of-domain corpus.

[2] It is located at http://www.fjoch.com/GIZA++.html.

3.2 Fertility Probability Interpolation

The fertility probability $n(\phi_i \mid e_i)$ describes the distribution of the number of words that e_i is aligned to. The interpolation model is shown in (9).

$$n(\phi_i \mid e_i) = \lambda_n \cdot n_I(\phi_i \mid e_i) + (1-\lambda_n) \cdot n_O(\phi_i \mid e_i) \quad (9)$$

Where, λ_n is a constant. This constant is obtained using a manually annotated held-out data set. In fact, we can also set the interpolation weight to be a function of the word e_i. From the word alignment results on the held-out set, we conclude that these two weighting schemes do not perform quite differently.

3.3 Distortion Probability Interpolation

The distortion probability describes the distribution of alignment positions. We separate it into two parts: one is the distortion probability in model 3, and the other is the distortion probability in model 4. The interpolation model for the distortion probability in model 3 is shown in (10). Since the distortion probability is irrelevant with any specific source or target words, we take λ_d as a constant. This constant is obtained using the held-out set.

$$\begin{aligned} d(j \mid a_j, l, m) = \lambda_d \cdot d_I(j \mid a_j, l, m) + \\ (1-\lambda_d) \cdot d_O(j \mid a_j, l, m) \end{aligned} \quad (10)$$

For the distortion probability in model 4, we use the same interpolation method and take the interpolation weight as a constant.

4 Translation Dictionary Acquisition

We use the translation dictionary trained from the training data to further improve the alignment results. When we train the bi-directional statistical word alignment models with the training data, we get two word alignment results for the training data. By taking the intersection of the two word alignment results, we build a new alignment set. The alignment links in this intersection set are extended by iteratively adding word alignment links into it as described in (Och and Ney, 2000). Based on the extended alignment links, we build a translation dictionary. In order to filter the noise caused by the error alignment links, we only retain those translation pairs whose log-likelihood ratio scores (Dunning, 1993) are above a threshold.

Based on the alignment results on the out-of-domain corpus, we build a translation dictionary D_1 filtered with a threshold δ_1. Based on the alignment results on a small-scale in-domain corpus, we build another translation dictionary D_2 filtered with a threshold δ_2.

After obtaining the two dictionaries, we combine two dictionaries through linearly interpolating the translation probabilities in the two dictionaries, which is shown in (11). The symbols f and e represent a single word or a phrase in the source and target languages. This differs from the translation probability in Equation (7), where these two symbols only represent single words.

$$p(f \mid e) = \lambda(e) \cdot p_I(f \mid e) + (1-\lambda(e)) \cdot p_O(f \mid e) \quad (11)$$

The interpolation weight is also a function of e. It is calculated as shown in (12)[3].

$$\lambda(e) = \frac{p_I(e)}{p_I(e) + p_O(e)} \quad (12)$$

$p_I(e)$ and $p_O(e)$ represent the relative frequencies of e in the in-domain corpus and out-of-domain corpus, respectively.

5 Adaptation Algorithm

The adaptation algorithms include two parts: a training algorithm and a testing algorithm. The training algorithm is shown in Figure 1.

After getting the two adaptation models and the translation dictionary, we apply them to the in-domain corpus to perform word alignment. Here we call it *testing algorithm*. The detailed algorithm is shown in Figure 2. For each sentence pair, there are two different word alignment results, from which the final alignment links are selected according to their translation probabilities in the dictionary D. The selection order is similar to that in the competitive linking algorithm (Melamed, 1997). The difference is that we allow many-to-one and one-to-many alignments.

6 Evaluation

We compare our method with four other methods. The first method is descried in (Wu and Wang, 2004). We call it "Result Adaptation (ResAdapt)".

[3] We also tried an adaptation coefficient to calculate the interpolation weight as in (8). However, the alignment results are not improved by using this coefficient for the dictionary.

```
Input: In-domain training data
       Out-of-domain training data

(1) Train two alignment models $M_I^{st}$ (source to target) and $M_I^{ts}$ (target to source) using the in-domain corpus.
(2) Train the other two alignment models $M_O^{st}$ and $M_O^{ts}$ using the out-of-domain corpus.
(3) Build an adaptation model $M^{st}$ based on $M_I^{st}$ and $M_O^{st}$, and build the other adaptation model $M^{ts}$ based on $M_I^{ts}$ and $M_O^{ts}$ using the interpolation methods described in section 3.
(4) Train a dictionary $D_1$ using the alignment results on the in-domain training data.
(5) Train another dictionary $D_2$ using the alignment results on the out-of-domain training data.
(6) Build an adaptation dictionary $D$ based on $D_1$ and $D_2$ using the interpolation method described in section 4.

Output: Alignment models $M^{st}$ and $M^{ts}$
        Translation dictionary $D$
```

Figure 1. Training Algorithm

```
Input: Alignment models $M^{st}$ and $M^{ts}$, translation dictionary $D$, and testing data

(1) Apply the adaptation model $M^{st}$ and $M^{ts}$ to the testing data to get two different alignment results.
(2) Select the alignment links with higher translation probability in the translation dictionary $D$.

Output: Alignment results on the testing data
```

Figure 2. Testing Algorithm

The second method "Gen+Spec" directly combines the out-of-domain corpus and the in-domain corpus as training data. The third method "Gen" only uses the out-of-domain corpus as training data. The fourth method "Spec" only uses the in-domain corpus as training data. For each of the last three methods, we first train bi-directional alignment models using the training data. Then we build a translation dictionary based on the alignment results on the training data and filter it using log-likelihood ratio as described in section 4.

6.1 Training and Testing Data

In this paper, we take English-Chinese word alignment as a case study. We use a sentence-aligned out-of-domain English-Chinese bilingual corpus, which includes 320,000 bilingual sentence pairs. The average length of the English sentences is 13.6 words while the average length of the Chinese sentences is 14.2 words.

We also use a sentence-aligned in-domain English-Chinese bilingual corpus (operation manuals for diagnostic ultrasound systems), which includes 5,862 bilingual sentence pairs. The average length of the English sentences is 12.8 words while the average length of the Chinese sentences is 11.8 words. From this domain-specific corpus, we randomly select 416 pairs as testing data. We also select 400 pairs to be manually annotated as held-out set (development set) to adjust parameters. The remained 5,046 pairs are used as domain-specific training data.

The Chinese sentences in both the training set and the testing set are automatically segmented into words. In order to exclude the effect of the segmentation errors on our alignment results, the segmentation errors in our testing set are post-corrected. The alignments in the testing set are manually annotated, which includes 3,166 alignment links. Among them, 504 alignment links include multiword units.

6.2 Evaluation Metrics

We use the same evaluation metrics as described in (Wu and Wang, 2004). If we use S_G to represent the set of alignment links identified by the proposed methods and S_C to denote the reference alignment set, the methods to calculate the precision, recall, f-measure, and alignment error rate (AER) are shown in Equation (13), (14), (15), and (16). It can be seen that the higher the f-measure is, the lower the alignment error rate is. Thus, we will only show precision, recall and AER scores in the evaluation results.

$$precision = \frac{|S_G \cap S_C|}{|S_G|} \quad (13)$$

$$recall = \frac{|S_G \cap S_C|}{|S_C|} \qquad (14)$$

$$fmeasure = \frac{2\times|S_G \cap S_C|}{|S_G|+|S_C|} \qquad (15)$$

$$AER = 1 - \frac{2\times|S_G \cap S_C|}{|S_G|+|S_C|} = 1 - fmeasure \qquad (16)$$

6.3 Evaluation Results

We use the held-out set described in section 6.1 to set the interpolation weights. The coefficient α in Equation (8) is set to 0.8, the interpolation weight λ_n in Equation (9) is set to 0.1, the interpolation weight λ_d in model 3 in Equation (10) is set to 0.1, and the interpolation weight λ_d in model 4 is set to 1. In addition, log-likelihood ratio score thresholds are set to $\delta_1 = 30$ and $\delta_2 = 25$. With these parameters, we get the lowest alignment error rate on the held-out set.

Using these parameters, we build two adaptation models and a translation dictionary on the training data, which are applied to the testing set. The evaluation results on our testing set are shown in Table 1. From the results, it can be seen that our approach performs the best among all of the methods, achieving the lowest alignment error rate. Compared with the method "ResAdapt", our method achieves a higher precision without loss of recall, resulting in an error rate reduction of 6.56%. Compared with the method "Gen+Spec", our method gets a higher recall, resulting in an error rate reduction of 17.43%. This indicates that our model adaptation method is very effective to alleviate the data-sparseness problem of domain-specific word alignment.

Method	Precision	Recall	AER
Ours	0.8490	0.7599	0.1980
ResAdapt	0.8198	0.7587	0.2119
Gen+Spec	0.8456	0.6905	0.2398
Gen	0.8589	0.6576	0.2551
Spec	0.8386	0.6731	0.2532

Table 1. Word Alignment Adaptation Results

The method that only uses the large-scale out-of-domain corpus as training data does not produce good result. The alignment error rate is almost the same as that of the method only using the in-domain corpus. In order to further analyze the result, we classify the alignment links into two classes: single word alignment links (SWA) and multiword alignment links (MWA). Single word alignment links only include one-to-one alignments. The multiword alignment links include those links in which there are multiword units in the source language or/and the target language. The results are shown in Table 2. From the results, it can be seen that the method "Spec" produces better results for multiword alignment while the method "Gen" produces better results for single word alignment. This indicates that the multiword alignment links mainly include the domain-specific words. Among the 504 multiword alignment links, about 60% of the links include domain-specific words. In Table 2, we also present the results of our method. Our method achieves the lowest error rate results on both single word alignment and multiword alignment.

Method	Precision	Recall	AER
Ours (SWA)	0.8703	0.8621	0.1338
Ours (MWA)	0.5635	0.2202	0.6833
Gen (SWA)	0.8816	0.7694	0.1783
Gen (MWA)	0.3366	0.0675	0.8876
Spec (SWA)	0.8710	0.7633	0.1864
Spec (MWA)	0.4760	0.1964	0.7219

Table 2. Single Word and Multiword Alignment Results

In order to further compare our method with the method described in (Wu and Wang, 2004). We do another experiment using almost the same-scale in-domain training corpus as described in (Wu and Wang, 2004). From the in-domain training corpus, we randomly select about 500 sentence pairs to build the smaller training set. The testing data is the same as shown in section 6.1. The evaluation results are shown in Table 3.

Method	Precision	Recall	AER
Ours	0.8424	0.7378	0.2134
ResAdapt	0.8027	0.7262	0.2375
Gen+Spec	0.8041	0.6857	0.2598

Table 3. Alignment Adaptation Results Using a Smaller In-Domain Corpus

Compared with the method "Gen+Spec", our method achieves an error rate reduction of 17.86%

while the method “ResAdapt” described in (Wu and Wang, 2004) only achieves an error rate reduction of 8.59%. Compared with the method “ResAdapt”, our method achieves an error rate reduction of 10.15%.

This result is different from that in (Wu and Wang, 2004), where their method achieved an error rate reduction of 21.96% as compared with the method “Gen+Spec”. The main reason is that the in-domain training corpus and testing corpus in this paper are different from those in (Wu and Wang, 2004). The training data and the testing data described in (Wu and Wang, 2004) are from a single manual. The data in our corpus are from several manuals describing how to use the diagnostic ultrasound systems.

In addition to the above evaluations, we also evaluate our model adaptation method using the "refined" combination in Och and Ney (2000) instead of the translation dictionary. Using the "refined" method to select the alignments produced by our model adaptation method (AER: 0.2371) still yields better result than directly combining out-of-domain and in-domain corpora as training data of the "refined" method (AER: 0.2290).

6.4 The Effect of In-Domain Corpus

In general, it is difficult to obtain large-scale in-domain bilingual corpus. For some domains, only a very small-scale bilingual sentence pairs are available. Thus, in order to analyze the effect of the size of in-domain corpus, we randomly select sentence pairs from the in-domain training corpus to generate five training sets. The numbers of sentence pairs in these five sets are 1,010, 2,020, 3,030, 4,040 and 5,046. For each training set, we use model 4 in section 2 to train an in-domain model. The out-of-domain corpus for the adaptation experiments and the testing set are the same as described in section 6.1.

# Sentence Pairs	Precision	Recall	AER
1010	0.8385	0.7394	0.2142
2020	0.8388	0.7514	0.2073
3030	0.8474	0.7558	0.2010
4040	0.8482	0.7555	0.2008
5046	0.8490	0.7599	0.1980

Table 4. Alignment Adaptation Results Using In-Domain Corpora of Different Sizes

# Sentence Pairs	Precision	Recall	AER
1010	0.8737	0.6642	0.2453
2020	0.8502	0.6804	0.2442
3030	0.8473	0.6874	0.2410
4040	0.8430	0.6917	0.2401
5046	0.8456	0.6905	0.2398

Table 5. Alignment Results Directly Combining Out-of-Domain and In-Domain Corpora

The results are shown in Table 4 and Table 5. Table 4 describes the alignment adaptation results using in-domain corpora of different sizes. Table 5 describes the alignment results by directly combining the out-of-domain corpus and the in-domain corpus of different sizes. From the results, it can be seen that the larger the size of in-domain corpus is, the smaller the alignment error rate is. However, when the number of the sentence pairs increase from 3030 to 5046, the error rate reduction in Table 4 is very small. This is because the contents in the specific domain are highly replicated. This also shows that increasing the domain-specific corpus does not obtain great improvement on the word alignment results. Comparing the results in Table 4 and Table 5, we find out that our adaptation method reduces the alignment error rate on all of the in-domain corpora of different sizes.

6.5 The Effect of Out-of-Domain Corpus

In order to further analyze the effect of the out-of-domain corpus on the adaptation results, we randomly select sentence pairs from the out-of-domain corpus to generate five sets. The numbers of sentence pairs in these five sets are 65,000, 130,000, 195,000, 260,000, and 320,000 (the entire out-of-domain corpus). In the adaptation experiments, we use the entire in-domain corpus (5046 sentence pairs). The adaptation results are shown in Table 6.

From the results in Table 6, it can be seen that the larger the size of out-of-domain corpus is, the smaller the alignment error rate is. However, when the number of the sentence pairs is more than 130,000, the error rate reduction is very small. This indicates that we do not need a very large bilingual out-of-domain corpus to improve domain-specific word alignment results.

# Sentence Pairs (k)	Precision	Recall	AER
65	0.8441	0.7284	0.2180
130	0.8479	0.7413	0.2090
195	0.8454	0.7461	0.2073
260	0.8426	0.7508	0.2059
320	0.8490	0.7599	0.1980

Table 6. Adaptation Alignment Results Using Out-of-Domain Corpora of Different Sizes

7 Conclusion

This paper proposes an approach to improve domain-specific word alignment through alignment model adaptation. Our approach first trains two alignment models with a large-scale out-of-domain corpus and a small-scale domain-specific corpus. Second, we build a new adaptation model by linearly interpolating these two models. Third, we apply the new model to the domain-specific corpus and improve the word alignment results. In addition, with the training data, an interpolated translation dictionary is built to select the word alignment links from different alignment results. Experimental results indicate that our approach achieves a precision of 84.90% and a recall of 75.99% for word alignment in a specific domain. Our method achieves a relative error rate reduction of 17.43% as compared with the method directly combining the out-of-domain corpus and the in-domain corpus as training data. It also achieves a relative error rate reduction of 6.56% as compared with the previous work in (Wu and Wang, 2004). In addition, when we train the model with a smaller-scale in-domain corpus as described in (Wu and Wang, 2004), our method achieves an error rate reduction of 10.15% as compared with the method in (Wu and Wang, 2004).

We also use in-domain corpora and out-of-domain corpora of different sizes to perform adaptation experiments. The experimental results show that our model adaptation method improves alignment results on in-domain corpora of different sizes. The experimental results also show that even a not very large out-of-domain corpus can help to improve the domain-specific word alignment through alignment model adaptation.

References

L. Ahrenberg, M. Merkel, M. Andersson. 1998. *A Simple Hybrid Aligner for Generating Lexical Correspondences in Parallel Tests.* In Proc. of ACL/COLING-1998, pp. 29-35.

Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. J. Och, D. Purdy, N. A. Smith, D. Yarowsky. 1999. *Statistical Machine Translation Final Report.* Johns Hopkins University Workshop.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, R. Mercer. 1993. *The Mathematics of Statistical Machine Translation: Parameter Estimation.* Computational Linguistics, 19(2): 263-311.

C. Cherry and D. Lin. 2003. *A Probability Model to Improve Word Alignment.* In Proc. of ACL-2003, pp. 88-95.

T. Dunning. 1993. *Accurate Methods for the Statistics of Surprise and Coincidence.* Computational Linguistics, 19(1): 61-74.

R. Iyer, M. Ostendorf, H. Gish. 1997. *Using Out-of-Domain Data to Improve In-Domain Language Models*. IEEE Signal Processing Letters, 221-223.

S. J. Ker and J. S. Chang. 1997. *A Class-based Approach to Word Alignment*. Computational Linguistics, 23(2): 313-343.

I. D. Melamed. 1997. *A Word-to-Word Model of Translational Equivalence*. In Proc. of ACL 1997, pp. 490-497.

F. J. Och and H. Ney. 2000. *Improved Statistical Alignment Models.* In Proc. of ACL-2000, pp. 440-447.

A. Peñas, F. Verdejo, J. Gonzalo. 2001. *Corpus-based Terminology Extraction Applied to Information Access.* In Proc. of the Corpus Linguistics 2001, vol. 13.

F. Smadja, K. R. McKeown, V. Hatzivassiloglou. 1996. *Translating Collocations for Bilingual Lexicons: a Statistical Approach.* Computational Linguistics, 22(1): 1-38.

D. Tufis and A. M. Barbu. 2002. *Lexical Token Alignment: Experiments, Results and Application.* In Proc. of LREC-2002, pp. 458-465.

D. Wu. 1997. *Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora.* Computational Linguistics, 23(3): 377-403.

H. Wu and H. Wang. 2004. *Improving Domain-Specific Word Alignment with a General Bilingual Corpus.* In R. E. Frederking and K. B. Taylor (Eds.), Machine Translation: From Real Users to Research: 6th conference of AMTA-2004, pp. 262-271.

Stochastic Lexicalized Inversion Transduction Grammar for Alignment

Hao Zhang and **Daniel Gildea**
Computer Science Department
University of Rochester
Rochester, NY 14627

Abstract

We present a version of Inversion Transduction Grammar where rule probabilities are lexicalized throughout the synchronous parse tree, along with pruning techniques for efficient training. Alignment results improve over unlexicalized ITG on short sentences for which full EM is feasible, but pruning seems to have a negative impact on longer sentences.

1 Introduction

The Inversion Transduction Grammar (ITG) of Wu (1997) is a syntactically motivated algorithm for producing word-level alignments of pairs of translationally equivalent sentences in two languages. The algorithm builds a synchronous parse tree for both sentences, and assumes that the trees have the same underlying structure but that the ordering of constituents may differ in the two languages.

This probabilistic, syntax-based approach has inspired much subsequent reasearch. Alshawi et al. (2000) use hierarchical finite-state transducers. In the tree-to-string model of Yamada and Knight (2001), a parse tree for one sentence of a translation pair is projected onto the other string. Melamed (2003) presents algorithms for synchronous parsing with more complex grammars, discussing how to parse grammars with greater than binary branching and lexicalization of synchronous grammars.

Despite being one of the earliest probabilistic syntax-based translation models, ITG remains state-of-the art. Zens and Ney (2003) found that the constraints of ITG were a better match to the decoding task than the heuristics used in the IBM decoder of Berger et al. (1996). Zhang and Gildea (2004) found ITG to outperform the tree-to-string model for word-level alignment, as measured against human gold-standard alignments. One explanation for this result is that, while a tree representation is helpful for modeling translation, the trees assigned by the traditional monolingual parsers (and the treebanks on which they are trained) may not be optimal for translation of a specific language pair. ITG has the advantage of being entirely data-driven – the trees are derived from an expectation maximization procedure given only the original strings as input.

In this paper, we extend ITG to condition the grammar production probabilities on lexical information throughout the tree. This model is reminiscent of lexicalization as used in modern statistical parsers, in that a unique head word is chosen for each constituent in the tree. It differs in that the head words are chosen through EM rather than deterministic rules. This approach is designed to retain the purely data-driven character of ITG, while giving the model more information to work with. By conditioning on lexical information, we expect the model to be able capture the same systematic differences in languages' grammars that motive the tree-to-string model, for example, SVO vs. SOV word order or prepositions vs. postpositions, but to be able to do so in a more fine-grained manner. The interaction between lexical information and word order also explains the higher performance of IBM model 4 over IBM model 3 for alignment.

We begin by presenting the probability model in the following section, detailing how we address issues of pruning and smoothing that lexicalization introduces. We present alignment results on a parallel Chinese-English corpus in Section 3.

2 Lexicalization of Inversion Transduction Grammars

An Inversion Transduction Grammar can generate pairs of sentences in two languages by recursively applying context-free bilingual production rules. Most work on ITG has focused on the 2-normal form, which consists of unary production rules that are responsible for generating word pairs:

$$X \to e/f$$

and binary production rules in two forms that are responsible for generating syntactic subtree pairs:

$$X \to [YZ]$$

and

$$X \to \langle YZ \rangle$$

The rules with square brackets enclosing the right hand side expand the left hand side symbol into the two symbols on the right hand side in the same order in the two languages, whereas the rules with pointed brackets expand the left hand side symbol into the two right hand side symbols in reverse order in the two languages.

One special case of ITG is the bracketing ITG that has only one nonterminal that instantiates exactly one straight rule and one inverted rule. The ITG we apply in our experiments has more structural labels than the primitive bracketing grammar: it has a start symbol S, a single preterminal C, and two intermediate nonterminals A and B used to ensure that only one parse can generate any given word-level alignment, as discussed by Wu (1997) and Zens and Ney (2003).

As an example, Figure 1 shows the alignment and the corresponding parse tree for the sentence pair *Je les vois / I see them* using the unambiguous bracketing ITG.

A stochastic ITG can be thought of as a stochastic CFG extended to the space of bitext. The independence assumptions typifying S-CFGs are also valid for S-ITGs. Therefore, the probability of an S-ITG parse is calculated as the product of the probabilities of all the instances of rules in the parse tree. For instance, the probability of the parse in Figure 1 is:

$$P(S \to A) \cdot P(A \to [CB]) \\ \cdot P(B \to \langle CC \rangle) \cdot P(C \to I/Je) \\ \cdot P(C \to see/vois) \cdot P(C \to them/les)$$

It is important to note that besides the bottom-level word-pairing rules, the other rules are all non-lexical, which means the structural alignment component of the model is not sensitive to the lexical contents of subtrees. Although the ITG model can effectively restrict the space of alignment to make polynomial time parsing algorithms possible, the preference for inverted or straight rules only passively reflect the need of bottom level word alignment. We are interested in investigating how much help it would be if we strengthen the structural alignment component by making the orientation choices dependent on the real lexical pairs that are passed up from the bottom.

The first step of lexicalization is to associate a lexical pair with each nonterminal. The head word pair generation rules are designed for this purpose:

$$X \to X(e/f)$$

The word pair e/f is representative of the lexical content of X in the two languages.

For binary rules, the mechanism of head selection is introduced. Now there are 4 forms of binary rules:

$$X(e/f) \to [Y(e/f)Z]$$
$$X(e/f) \to [YZ(e/f)]$$
$$X(e/f) \to \langle Y(e/f)Z \rangle$$
$$X(e/f) \to \langle YZ(e/f) \rangle$$

determined by the four possible combinations of head selections (Y or Z) and orientation selections (straight or inverted).

The rules for generating lexical pairs at the leaves of the tree are now predetermined:

$$X(e/f) \to e/f$$

Putting them all together, we are able to derive a lexicalized bilingual parse tree for a given sentence pair. In Figure 2, the example in Figure 1 is revisited. The probability of the lexicalized parse is:

$$P(S \to S(see/vois)) \\ \cdot P(S(see/vois) \to A(see/vois)) \\ \cdot P(A(see/vois) \to [CB(see/vois)]) \\ \cdot P(C \to C(I/Je))$$

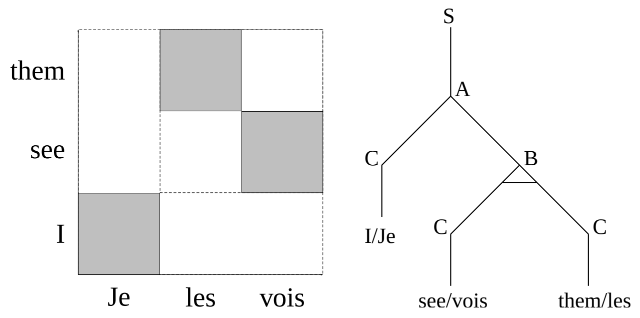


Figure 1: ITG Example

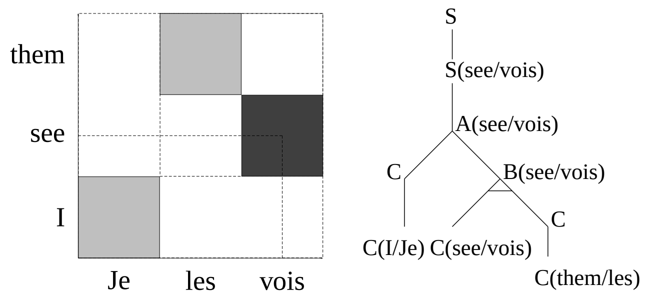


Figure 2: Lexicalized ITG Example. *see/vois* is the headword of both the 2x2 cell and the entire alignment.

$$\cdot P(B(see/vois) \to \langle C(see/vois)C\rangle)$$
$$\cdot P(C \to C(them/les))$$

The factors of the product are ordered to show the generative process of the most probable parse. Starting from the start symbol S, we first choose the head word pair for S, which is *see/vois* in the example. Then, we recursively expand the lexicalized head constituents using the lexicalized structural rules. Since we are only lexicalizing rather than bilexicalizing the rules, the non-head constituents need to be lexicalized using head generation rules so that the top-down generation process can proceed in all branches. By doing so, word pairs can appear at all levels of the final parse tree in contrast with the unlexicalized parse tree in which the word pairs are generated only at the bottom.

The binary rules are lexicalized rather than bilexicalized.[1] This is a trade-off between complexity and expressiveness. After our lexicalization, the number of lexical rules, thus the number of parameters in the statistical model, is still at the order of $O(|V||T|)$, where $|V|$ and $|T|$ are the vocabulary sizes of the two languages.

[1] In a sense our rules are bilexicalized in that they condition on words from both languages; however they do not capture head-modifier relations within a language.

2.1 Parsing

Given a bilingual sentence pair, a synchronous parse can be built using a two-dimensional extension of chart parsing, where chart items are indexed by their nonterminal X, head word pair e/f if specified, beginning and ending positions l, m in the source language string, and beginning and ending positions i, j in the target language string. For Expectation Maximization training, we compute lexicalized inside probabilities $\beta(X(e/f), l, m, i, j)$, as well as unlexicalized inside probabilities $\beta(X, l, m, i, j)$, from the bottom up as outlined in Algorithm 1.

The algorithm has a complexity of $O(N_s^4 N_t^4)$, where N_s and N_t are the lengths of source and target sentences respectively. The complexity of parsing for an unlexicalized ITG is $O(N_s^3 N_t^3)$. Lexicalization introduces an additional factor of $O(N_s N_t)$, caused by the choice of headwords e and f in the pseudocode.

Assuming that the lengths of the source and target sentences are proportional, the algorithm has a complexity of $O(n^8)$, where n is the average length of the source and target sentences.

Algorithm 1 LexicalizedITG(s, t)

```
for all $l, m$ such that $0 \leq l \leq m \leq N_s$ do
  for all $i, j$ such that $0 \leq i \leq j \leq N_t$ do
    for all $e \in \{e_{l+1} \dots e_m\}$ do
      for all $f \in \{f_{i+1} \dots f_j\}$ do
        for all $n$ such that $l \leq n \leq m$ do
          for all $k$ such that $i \leq k \leq j$ do
            for all rules $X \to YZ \in G$ do
              $\beta(X(e/f), l, m, i, j)$ +=
              $\triangleright$ straight rule, where $Y$ is head
                $P([Y(e/f)Z] \mid X(e/f)) \cdot \beta(Y(e/f), l, n, i, k) \cdot \beta(Z, n, m, k, j)$
              $\triangleright$ inverted rule, where $Y$ is head
               $+ P(\langle Y(e/f)Z \rangle \mid X(e/f)) \cdot \beta(Y(e/f), n, m, i, k) \cdot \beta(Z, l, n, k, j)$
              $\triangleright$ straight rule, where $Z$ is head
               $+ P([YZ(e/f)] \mid X(e/f)) \cdot \beta(Y, l, n, i, k) \cdot \beta(Z(e/f), n, m, k, j)$
              $\triangleright$ inverted rule, where $Z$ is head
               $+ P(\langle YZ(e/f) \rangle \mid X(e/f)) \cdot \beta(Y, n, m, i, k) \cdot \beta(Z(e/f), l, n, k, j)$
            end for
          end for
        end for
        $\triangleright$ word pair generation rule
        $\beta(X, l, m, i, j)$ += $P(X(e/f) \mid X) \cdot \beta(X(e/f), l, m, i, j)$
      end for
    end for
  end for
end for
```

2.2 Pruning

We need to further restrict the space of alignments spanned by the source and target strings to make the algorithm feasible. Our technique involves computing an estimate of how likely each of the n^4 cells in the chart is before considering all ways of building the cell by combining smaller subcells. Our figure of merit for a cell involves an estimate of both the inside probability of the cell (how likely the words within the box in both dimensions are to align) and the outside probability (how likely the words outside the box in both dimensions are to align). In including an estimate of the outside probability, our technique is related to A* methods for monolingual parsing (Klein and Manning, 2003), although our estimate is not guaranteed to be lower than complete outside probabity assigned by ITG. Figure 3(a) displays the tic-tac-toe pattern for the inside and outside components of a particular cell. We use IBM Model 1 as our estimate of both the inside and outside probabilities. In the Model 1 estimate of the outside probability, source and target words can align using any combination of points from the four outside corners of the tic-tac-toe pattern. Thus in Figure 3(a), there is one solid cell (corresponding to the Model 1 Viterbi alignment) in each column, falling either in the upper or lower outside shaded corner. This can be also be thought of as squeezing together the four outside corners, creating a new cell whose probability is estimated using IBM Model 1. Mathematically, our figure of merit for the cell (l, m, i, j) is a product of the inside Model 1 probability and the outside Model 1 probability:

$$P(\mathbf{f}_{(i,j)} \mid \mathbf{e}_{(l,m)}) \cdot P(\mathbf{f}_{\overline{(i,j)}} \mid \mathbf{e}_{\overline{(l,m)}}) \tag{1}$$
$$= \lambda_{|(l,m)|,|(i,j)|} \prod_{t \in (i,j)} \sum_{s \in \{0,(l,m)\}} t(f_t \mid e_s)$$
$$\cdot \lambda_{|\overline{(l,m)}|,|\overline{(i,j)}|} \prod_{t \in \overline{(i,j)}} \sum_{s \in \{0,\overline{(l,m)}\}} t(f_t \mid e_s)$$

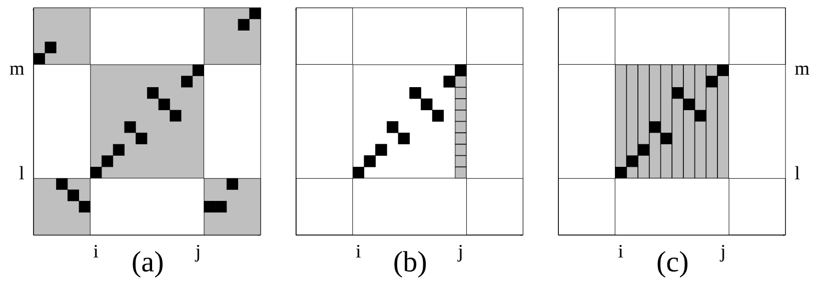


Figure 3: The tic-tac-toe figure of merit used for pruning bitext cells. The shaded regions in (a) show alignments included in the figure of merit for bitext cell (l, m, i, j) (Equation 1); solid black cells show the Model 1 Viterbi alignment within the shaded area. (b) shows how to compute the inside probability of a unit-width cell by combining basic cells (Equation 2), and (c) shows how to compute the inside probability of any cell by combining unit-width cells (Equation 3).

where $\overline{(l,m)}$ and $\overline{(i,j)}$ represent the complementary spans in the two languages. λ_{L_1,L_2} is the probability of any word alignment template for a pair of L_1-word source string and L_2-word target string, which we model as a uniform distribution of word-for-word alignment patterns after a Poisson distribution of target string's possible lengths, following Brown et al. (1993). As an alternative, the $\sum$ operator can be replaced by the $\max$ operator as the inside operator over the translation probabilities above, meaning that we use the Model 1 Viterbi probability as our estimate, rather than the total Model 1 probability.[2]

A naïve implementation would take $O(n^6)$ steps of computation, because there are $O(n^4)$ cells, each of which takes $O(n^2)$ steps to compute its Model 1 probability. Fortunately, we can exploit the recursive nature of the cells. Let $\text{INS}(l, m, i, j)$ denote the major factor of our Model 1 estimate of a cell's inside probability, $\prod_{t\in(i,j)} \sum_{s\in\{0,(l,m)\}} t(f_t \mid e_s)$. It turns out that one can compute cells of width one $(i = j)$ in constant time from a cell of equal width and lower height:

$$\begin{aligned}\text{INS}(l, m, j, j) &= \prod_{t\in(j,j)} \sum_{s\in\{0,(l,m)\}} t(f_t \mid e_s)\\ &= \sum_{s\in\{0,(l,m)\}} t(f_j \mid e_s)\\ &= \text{INS}(l, m-1, j, j)\\ &\quad + t(f_j \mid e_m) \qquad (2)\end{aligned}$$

Similarly, one can compute cells of width greater than one by combining a cell of one smaller width with a cell of width one:

$$\begin{aligned}\text{INS}(l, m, i, j) &= \prod_{t\in(i,j)} \sum_{s\in\{0,(l,m)\}} t(f_t \mid e_s)\\ &= \prod_{t\in(i,j)} \text{INS}(l, m, t, t)\\ &= \text{INS}(l, m, i, j-1)\\ &\quad \cdot \text{INS}(l, m, j, j) \qquad (3)\end{aligned}$$

Figure 3(b) and (c) illustrate the inductive computation indicated by the two equations. Each of the $O(n^4)$ inductive steps takes one additive or multiplicative computation. A similar dynammic programing technique can be used to efficiently compute the outside component of the figure of merit. Hence, the algorithm takes just $O(n^4)$ steps to compute the figure of merit for all cells in the chart.

Once the cells have been scored, there can be many ways of pruning. In our experiments, we applied beam ratio pruning to each individual bucket of cells sharing a common source substring. We prune cells whose probability is lower than a fixed ratio below the best cell for the same source substring. As a result, at least one cell will be kept for each source substring. We safely pruned more than 70% of cells using 10^{-5} as the beam ratio for sentences up to 25 words. Note that this pruning technique is applicable to both the lexicalized ITG and the conventional ITG.

In addition to pruning based on the figure of merit described above, we use top-k pruning to limit the number of hypotheses retained for each cell. This is necessary for lexicalized ITG because the number of distinct hypotheses in the two-dimensional ITG

[2]The experimental difference of the two alternatives was small. For our results, we used the $\max$ version.

chart has increased to $O(N_s^3 N_t^3)$ from $O(N_s^2 N_t^2)$ due to the choice one of $O(N_s)$ source language words and one of $O(N_t)$ target language words as the head. We keep only the top-k lexicalized items for a given chart cell of a certain nonterminal Y contained in the cell l, m, i, j. Thus the additional complexity of $O(N_s N_t)$ will be replaced by a constant factor.

The two pruning techniques can work for both the computation of expected counts during the training process and for the Viterbi-style algorithm for extracting the most probable parse after training. However, if we initialize EM from a uniform distribution, all probabilties are equal on the first iteration, giving us no basis to make pruning decisions. So, in our experiments, we initialize the head generation probabilities of the form $P(X(e/f) \mid X)$ to be the same as $P(e/f \mid C)$ from the result of the unlexicalized ITG training.

2.3 Smoothing

Even though we have controlled the number of parameters of the model to be at the magnitude of $O(|V||T|)$, the problem of data sparseness still renders a smoothing method necessary. We use backing off smoothing as the solution. The probabilities of the unary head generation rules are in the form of $P(X(e/f) \mid X)$. We simply back them off to the uniform distribution. The probabilities of the binary rules, which are conditioned on lexicalized nonterminals, however, need to be backed off to the probabilities of generalized rules in the following forms:

$$P([Y(*)Z] \mid X(*))$$

$$P([YZ(*)] \mid X(*))$$

$$P(\langle Y(*)Z\rangle \mid X(*))$$

$$P(\langle YZ(*)\rangle \mid X(*))$$

where $*$ stands for any lexical pair. For instance,

$$\begin{aligned} P([Y(e/f)Z] \mid X(e/f)) = \\ &(1-\lambda)P_{EM}([Y(e/f)Z] \mid X(e/f)) \\ &+ \lambda P([Y(*)Z] \mid X(*)) \end{aligned}$$

where

$$\lambda = 1/(1 + Expected\_Counts(X(e/f)))$$

The more often $X(e/f)$ occurred, the more reliable are the estimated conditional probabilities with the condition part being $X(e/f)$.

3 Experiments

We trained both the unlexicalized and the lexicalized ITGs on a parallel corpus of Chinese-English newswire text. The Chinese data were automatically segmented into tokens, and English capitalization was retained. We replaced words occurring only once with an unknown word token, resulting in a Chinese vocabulary of 23,783 words and an English vocabulary of 27,075 words.

In the first experiment, we restricted ourselves to sentences of no more than 15 words in either language, resulting in a training corpus of 6,984 sentence pairs with a total of 66,681 Chinese words and 74,651 English words. In this experiment, we didn't apply the pruning techniques for the lexicalized ITG.

In the second experiment, we enabled the pruning techniques for the LITG with the beam ratio for the tic-tac-toe pruning as 10^{-5} and the number k for the top-k pruning as 25. We ran the experiments on sentences up to 25 words long in both languages. The resulting training corpus had 18,773 sentence pairs with a total of 276,113 Chinese words and 315,415 English words.

We evaluate our translation models in terms of agreement with human-annotated word-level alignments between the sentence pairs. For scoring the Viterbi alignments of each system against gold-standard annotated alignments, we use the alignment error rate (AER) of Och and Ney (2000), which measures agreement at the level of pairs of words:

$$AER = 1 - \frac{|A \cap G_P| + |A \cap G_S|}{|A| + |G_S|}$$

where A is the set of word pairs aligned by the automatic system, G_S is the set marked in the gold standard as "sure", and G_P is the set marked as "possible" (including the "sure" pairs). In our Chinese-English data, only one type of alignment was marked, meaning that $G_P = G_S$.

In our hand-aligned data, 20 sentence pairs are less than or equal to 15 words in both languages, and were used as the test set for the first experiment, and 47 sentence pairs are no longer than 25 words in either language and were used to evaluate the pruned

	Precision	*Recall*	*Alignment Error Rate*
IBM Model 1	.59	.37	.54
IBM Model 4	.63	.43	.49
ITG	.62	.47	.46
Lexicalized ITG	.66	.50	.43

Table 1: Alignment results on Chinese-English corpus ($\leq$ 15 words on both sides). Full ITG vs. Full LITG

	Precision	*Recall*	*Alignment Error Rate*
IBM Model 1	.56	.42	.52
IBM Model 4	.67	.43	.47
ITG	.68	.52	.40
Lexicalized ITG	.69	.51	.41

Table 2: Alignment results on Chinese-English corpus ($\leq$ 25 words on both sides). Full ITG vs. Pruned LITG

LITG against the unlexicalized ITG.

A separate development set of hand-aligned sentence pairs was used to control overfitting. The subset of up to 15 words in both languages was used for cross-validating in the first experiment. The subset of up to 25 words in both languages was used for the same purpose in the second experiment.

Table 1 compares results using the full (unpruned) model of unlexicalized ITG with the full model of lexicalized ITG.

The two models were initialized from uniform distributions for all rules and were trained until AER began to rise on our held-out cross-validation data, which turned out to be 4 iterations for ITG and 3 iterations for LITG.

The results from the second experiment are shown in Table 2. The performance of the full model of unlexicalized ITG is compared with the pruned model of lexicalized ITG using more training data and evaluation data.

Under the same check condition, we trained ITG for 3 iterations and the pruned LITG for 1 iteration.

For comparison, we also included the results from IBM Model 1 and Model 4. The numbers of iterations for the training of the IBM models were chosen to be the turning points of AER changing on the cross-validation data.

4 Discussion

As shown by the numbers in Table 1, the full lexicalized model produced promising alignment results on sentence pairs that have no more than 15 words on both sides. However, due to its prohibitive $O(n^8)$ computational complexity, our C++ implementation of the unpruned lexicalized model took more than 500 CPU hours, which were distributed over multiple machines, to finish one iteration of training. The number of CPU hours would increase to a point that is unacceptable if we doubled the average sentence length. Some type of pruning is a must-have. Our pruned version of LITG controlled the running time for one iteration to be less than 1200 CPU hours, despite the fact that both the number of sentences and the average length of sentences were more than doubled. To verify the safety of the tic-tac-toe pruning technique, we applied it to the unlexicalized ITG using the same beam ratio (10^{-5}) and found that the AER on the test data was not changed. However, whether or not the top-k lexical head pruning technique is equally safe remains a question. One noticeable implication of this technique for training is the reliance on initial probabilities of lexical pairs that are discriminative enough. The comparison of results for ITG and LITG in Table 2 and the fact that AER began to rise after only one iteration of training seem to indicate that keeping few distinct lexical heads caused convergence on a suboptimal set of parameters, leading to a form of overfitting. In contrast, overfitting did not seem to be a problem for LITG in the unpruned experiment of Table 1, despite the much larger number of parameters for LITG than for ITG and the smaller training set.

We also want to point out that for a pair of long sentences, it would be hard to reflect the inherent bilingual syntactic structure using the lexicalized binary bracketing parse tree. In Figure 2, $A(see/vois)$ echoes $IP(see/vois)$ and $B(see/vois)$ echoes $VP(see/vois)$ so that it means $IP(see/vois)$ is not inverted from English to French but its right child $VP(see/vois)$ is inverted. However, for longer sentences with more than 5 levels of bracketing and the same lexicalized nonterminal repeatedly appearing at different levels, the correspondences would become less linguistically plausible. We think the limitations of the bracketing grammar are another reason for not being able to improve the AER of longer sentence pairs after lexicalization.

The space of alignments that is to be considered by LITG is exactly the space considered by ITG since the structural rules shared by them define the alignment space. The lexicalized ITG is designed to be more sensitive to the lexical influence on the choices of inversions so that it can find better alignments. Wu (1997) demonstrated that for pairs of sentences that are less than 16 words, the ITG alignment space has a good coverage over all possibilities. Hence, it's reasonable to see a better chance of improving the alignment result for sentences less than 16 words.

5 Conclusion

We presented the formal description of a Stochastic Lexicalized Inversion Transduction Grammar with its EM training procedure, and proposed specially designed pruning and smoothing techniques. The experiments on a parallel corpus of Chinese and English showed that lexicalization helped for aligning sentences of up to 15 words on both sides. The pruning and the limitations of the bracketing grammar may be the reasons that the result on sentences of up to 25 words on both sides is not better than that of the unlexicalized ITG.

Acknowledgments We are very grateful to Rebecca Hwa for assistance with the Chinese-English data, to Kevin Knight and Daniel Marcu for their feedback, and to the authors of GIZA. This work was partially supported by NSF ITR IIS-09325646 and NSF ITR IIS-0428020.

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.

Adam Berger, Peter Brown, Stephen Della Pietra, Vincent Della Pietra, J. R. Fillett, Andrew Kehler, and Robert Mercer. 1996. Language translation apparatus and method of using context-based tanslation models. United States patent 5,510,981.

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.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*.

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the 2003 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-03)*, Edmonton.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Conference of the Association for Computational Linguistics (ACL-00)*, pages 440–447, Hong Kong, October.

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 39th Annual Conference of the Association for Computational Linguistics (ACL-01)*, Toulouse, France.

Richard Zens and Hermann Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.

Hao Zhang and Daniel Gildea. 2004. Syntax-based alignment: Supervised or unsupervised? In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, Geneva, Switzerland, August.

Multi-Field Information Extraction and Cross-Document Fusion

Gideon S. Mann and **David Yarowsky**
Department of Computer Science
The Johns Hopkins University
Baltimore, MD 21218 USA
{gsm,yarowsky}@cs.jhu.edu

Abstract

In this paper, we examine the task of extracting a set of biographic facts about target individuals from a collection of Web pages. We automatically annotate training text with positive and negative examples of fact extractions and train Rote, Naïve Bayes, and Conditional Random Field extraction models for fact extraction from individual Web pages. We then propose and evaluate methods for fusing the extracted information across documents to return a consensus answer. A novel cross-field bootstrapping method leverages data interdependencies to yield improved performance.

1 Introduction

Much recent statistical information extraction research has applied graphical models to extract information from one particular document after training on a large corpus of annotated data (Leek, 1997; Freitag and McCallum, 1999).[1] Such systems are widely applicable, yet there remain many information extraction tasks that are not readily amenable to these methods. Annotated data required for training statistical extraction systems is sometimes unavailable, while there are examples of the desired information. Further, the goal may be to find a few interrelated pieces of information that are stated multiple times in a set of documents.

Here, we investigate one task that meets the above criteria. Given the name of a celebrity such as "Frank Zappa", our goal is to extract a set of biographic facts (e.g., birthdate, birth place and occupation) about that person from documents on the Web.

First, we describe a general method of automatic annotation for training from positive and negative examples and use the method to train Rote, Naïve Bayes, and Conditional Random Field models (Section 2). We then examine how multiple extractions can be combined to form one consensus answer (Section 3). We compare fusion methods and show that frequency voting outperforms the single highest confidence answer by an average of 11% across the various extractors. Increasing the number of retrieved documents boosts the overall system accuracy as additional documents which mention the individual in question lead to higher recall. This improved recall more than compensates for a loss in per-extraction precision from these additional documents. Next, we present a method for cross-field bootstrapping (Section 4) which improves per-field accuracy by 7%. We demonstrate that a small training set with only the most relevant documents can be as effective as a larger training set with additional, less relevant documents (Section 5).

2 Training by Automatic Annotation

Typically, statistical extraction systems (such as HMMs and CRFs) are trained using hand-annotated data. Annotating the necessary data by hand is time-consuming and brittle, since it may require large-scale re-annotation when the annotation scheme changes. For the special case of Rote extractors, a more attractive alternative has been proposed by Brin (1998), Agichtein and Gravano (2000), and Ravichandran and Hovy (2002).

[1] Alternatively, Riloff (1996) trains on in-domain and out-of-domain texts and then has a human filtering step. Huffman (1995) proposes a method to train a different type of extraction system by example.

Essentially, for any text snippet of the form $A_1pA_2qA_3$, these systems estimate the probability that a relationship $r(p,q)$ holds between entities p and q, given the interstitial context, as[2]

$$\begin{aligned}P(r(p,q) \mid pA_2q) &= P(r(p,q) \mid pA_2q) \\ &= \frac{\sum_{x,y \in T} c(xA_2y)}{\sum_x c(xA_2)}\end{aligned}$$

That is, the probability of a relationship $r(p,q)$ is the number of times that pattern xA_2y predicts any relationship $r(x,y)$ in the training set T. $c(.)$ is the count. We will refer to x as the **hook**[3] and y as the **target**. In this paper, the hook is always an individual. Training a Rote extractor is straightforward given a set T of example relationships $r(x,y)$. For each hook, download a separate set of relevant documents (a **hook corpus**, D_x) from the Web.[4] Then for any particular pattern A_2 and an element x, count how often the pattern xA_2 predicts y and how often it retrieves a spurious $\bar{y}$.[5]

This annotation method extends to training other statistical models with positive examples, for example a Naïve Bayes (**NB**) unigram model. In this model, instead of looking for an exact A_2 pattern as above, each individual word in the pattern A_2 is used to predict the presence of a relationship.

$$\begin{aligned}&P(r(p,q) \mid pA_2q) \\ &\qquad \propto P(pA_2q \mid r(p,q))P(r(p,q)) \\ &\qquad = P(A_2 \mid r(p,q)) \\ &\qquad = \prod_{a \in A_2} P(a \mid r(p,q))\end{aligned}$$

We perform add-lambda smoothing for out-of-vocabulary words and thus assign a positive probability to any sequence. As before, a set of relevant documents is downloaded for each particular hook. Then every hook and target is annotated. From that markup, we can pick out the interstitial A_2 patterns and calculate the necessary probabilities.

Since the NB model assigns a positive probability to every sequence, we need to pick out likely targets from those proposed by the NB extractor. We construct a **background model** which is a basic unigram language model, $P(A_2) = \prod_{a \in A_2} P(a)$. We then pick targets chosen by the confidence estimate

$$C^{\textbf{NB}}(q) = log \frac{P(A_2 \mid r(p,q))}{P(A_2)}$$

However, this confidence estimate does not work-well in our dataset.

We propose to use negative examples to estimate $P(A_2 \mid \bar{r}(p,q))$[6] as well as $P(A_2 \mid r(p,q))$. For each relationship, we define the **target set** E_r to be all potential targets and model it using regular expressions.[7] In training, for each relationship $r(p,q)$, we markup the hook p, the target q, and all **spurious targets** ($\bar{q} \in \{E_r - q\}$) which provide negative examples. Targets can then be chosen with the following confidence estimate

$$C^{\textbf{NB+E}}(q) = log \frac{P(A_2 \mid r(p,q))}{P(A_2 \mid \bar{r}(p,q))}$$

We call this **NB+E** in the following experiments.

The above process describes a general method for automatically annotating a corpus with positive and negative examples, and this corpus can be used to train statistical models that rely on annotated data.[8] In this paper, we test automatic annotation using Conditional Random Fields (**CRFs**) (Lafferty et al., 2001) which have achieved high performance for information extraction. CRFs are undirected graphical models that estimate the conditional probability of a state sequence given an output sequence

$$P(\mathbf{s} \mid \mathbf{o}) = \frac{1}{Z} \exp\left(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t)\right)$$

[2]The above Rote models also condition on the preceding and trailing words, for simplicity we only model interstitial words A_2.

[3]Following (Ravichandran and Hovy, 2002).

[4]In the following experiments we assume that there is one main object of interest p, for whom we want to find certain pieces of information $r(p,q)$, where r denotes the type of relationship (e.g., birthday) and q is a value (e.g., May 20th). We require one hook corpus for each hook, not a separate one for each relationship.

[5]Having a **functional constraint** $\forall \bar{q} \neq q, \bar{r}(p,\bar{q})$ makes this estimate much more reliable, but it is possible to use this method of estimation even when this constraint does not hold.

[6]$\bar{r}$ stands in for all other possible relationships (including no relationship) between p and q. $P(A_2 \mid \bar{r}(p,q))$ is estimated as $P(A_2 \mid r(p,q))$ is, except with spurious targets.

[7]e.g., $E_{birthyear} = \{\backslash d \backslash d \backslash d \backslash d\}$. This is the only source of human knowledge put into the system and required only around 4 hours of effort, less effort than annotating an entire corpus or writing information extraction rules.

[8]This corpus markup gives automatic annotation that yields noisier training data than manual annotation would.

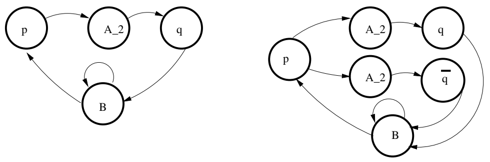


Figure 1: CRF state-transition graphs for extracting a relationship $r(p, q)$ from a sentence pA_2q. Left: **CRF** Extraction with a background model (B). Right: **CRF+E** As before but with spurious target prediction ($pA_2\bar{q}$).

We use the Mallet system (McCallum, 2002) for training and evaluation of the CRFs. In order to examine the improvement by using negative examples, we train CRFs with two topologies (Figure 1). The first, **CRF**, models the target relationship and background sequences and is trained on a corpus where targets (positive examples) are annotated. The second, **CRF+E**, models the target relationship, spurious targets and background sequences, and it is trained on a corpus where targets (positive examples) as well as spurious targets (negative examples) are annotated.

Experimental Results

To test the performance of the different extractors, we collected a set of 152 semi-structured mini-biographies from an online site (www.infoplease.com), and used simple rules to extract a biographic fact database of birthday and month (henceforth birthday), birth year, occupation, birth place, and year of death (when applicable). An example of the data can be found in Table 1. In our system, we normalized birthdays, and performed capitalization normalization for the remaining fields. We did no further normalization, such as normalizing state names to their two letter acronyms (e.g., California $\rightarrow$ CA). Fifteen names were set aside as training data, and the rest were used for testing. For each name, 150 documents were downloaded from Google to serve as the hook corpus for either training or testing.[9]

In training, we automatically annotated documents using people in the training set as hooks, and in testing, tried to get targets that exactly matched what was present in the database. This is a very strict method of evaluation for three reasons. First, since the facts were automatically collected, they contain errors and thus the system is tested against wrong answers.[10] Second, the extractors might have retrieved information that was simply not present in the database but nevertheless correct (e.g., someone's occupation might be listed as writer and the retrieved occupation might be novelist). Third, since the retrieved targets were not normalized, there system may have retrieved targets that were correct but were not recognized (e.g., the database birthplace is New York, and the system retrieves NY).

[9]Name polyreference, along with ranking errors, result in the retrieval of undesired documents.

	Aaron Neville	Frank Zappa
Birthday	January 24	December 21
Birth year	1941	1940
Occupation	Singer	Musician
Birthplace	New Orleans	Baltimore,Maryland
Year of Death	-	1993

Table 1: Two of 152 entries in the Biographic Database. Each entry contains incomplete information about various celebrities. Here, Aaron Neville's birth state is missing, and Frank Zappa could be equally well described as a guitarist or rock-star.

In testing, we rejected candidate targets that were not present in our target set models E_r. In some cases, this resulted in the system being unable to find the correct target for a particular relationship, since it was not in the target set.

Before fusion (Section 3), we gathered all the facts extracted by the system and graded them in isolation. We present the per-extraction **precision**

$$\textit{Pre-Fusion Precision} = \frac{\textit{\# Correct Extracted Targets}}{\textit{\# Total Extracted Targets}}$$

We also present the **pseudo-recall**, which is the average number of times per person a correct target was extracted. It is difficult to calculate true recall without manual annotation of the entire corpus, since it cannot be known for certain how many times the document set contains the desired information.[11]

$$\textit{Pre-Fusion Pseudo-Recall} = \frac{\textit{\# Correct Extracted Targets}}{\#People}$$

The precision of each of the various extraction methods is listed in Table 2. The data show that on average the Rote method has the best precision,

[10]These deficiencies in testing also have implications for training, since the models will be trained on annotated data that has errors. The phenomenon of missing and inaccurate data was most prevalent for occupation and birthplace relationships, though it was observed for other relationships as well.

[11]It is insufficient to count all text matches as instances that the system should extract. To obtain the true recall, it is necessary to decide whether each sentence contains the desired relationship, even in cases where the information is not what the biographies have listed.

	Birthday	Birth year	Occupation	Birthplace	Year of Death	Avg.
Rote	.789	.355	.305	.510	.527	.497
NB+E	.423	.361	.255	.217	.088	.269
CRF	.509	.342	.219	.139	.267	.295
CRF+E	.680	.654	.246	.357	.314	.450

Table 2: Pre-Fusion Precision of extracted facts for various extraction systems, trained on 15 people each with 150 documents, and tested on 137 people each with 150 documents.

	Birthday	Birth year	Occupation	Birthplace	Year of Death	Avg.
Rote	4.8	1.9	1.5	1.0	0.1	1.9
NB+E	9.6	11.5	20.3	11.3	0.7	10.9
CRF	3.0	16.3	31.1	10.7	3.2	12.9
CRF+E	6.8	9.9	3.2	3.6	1.4	5.0

Table 3: Pre-Fusion Pseudo-Recall of extract facts with the identical training/testing set-up as above.

while the NB+E extractor has the worst. Training the CRF with negative examples (CRF+E) gave better precision in extracted information then training it without negative examples. Table 3 lists the pseudo-recall or average number of correctly extracted targets per person. The results illustrate that the Rote has the worst pseudo-recall, and the plain CRF, trained without negative examples, has the best pseudo-recall.

To test how the extraction precision changes as more documents are retrieved from the ranked results from Google, we created retrieval sets of 1, 5, 15, 30, 75, and 150 documents per person and repeated the above experiments with the CRF+E extractor. The data in Figure 2 suggest that there is a gradual drop in extraction precision throughout the corpus, which may be caused by the fact that documents further down the retrieved list are less relevant, and therefore less likely to contain the relevant biographic data.

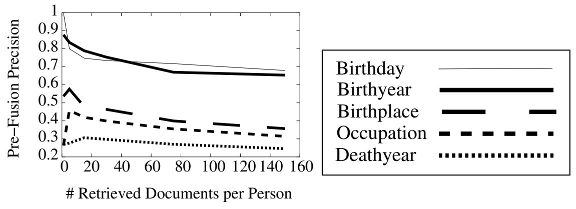


Figure 2: As more documents are retrieved per person, pre-fusion precision drops.

However, even though the extractor's precision drops, the data in Figure 3 indicate that there continue to be instances of the relevant biographic data.

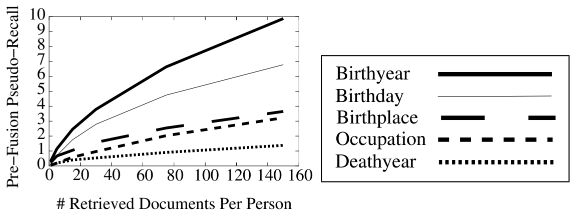


Figure 3: Pre-fusion pseudo-recall increases as more documents are added.

3 Cross-Document Information Fusion

The per-extraction performance was presented in Section 2, but the final task is to find the single correct target for each person.[12] In this section, we examine two basic methodologies for combining candidate targets. Masterson and Kushmerick (2003) propose **Best** which gives each candidate a score equal to its highest confidence extraction: **Best**$(x) = \operatorname{argmax}_{x} C(x)$.[13] We further consider **Voting**, which counts the number of times each candidate x was extracted: **Vote**$(x) = |C(x) > 0|$. Each of these methods ranks the candidate targets by score and chooses the top-ranked one.

The experimental setup used in the fusion experiments was the same as before: training on 15 people, and testing on 137 people. However, the post-fusion evaluation differs from the pre-fusion evaluation. After fusion, the system returns one consensus target for each person and thus the evaluation is on the **accuracy** of those targets. That is, missing tar-

[12]This is a simplifying assumption, since there are many cases where there might exist multiple possible values, e.g., a person may be both a writer and a musician.

[13]$C(x)$ is either the confidence estimate (NB+E) or the probability score (Rote,CRF,CRF+E).

	Best	Vote
Rote	.364	.450
NB+E	.385	.588
CRF	.513	.624
CRF+E	.650	.678

Table 4: Average Accuracy of the Highest Confidence (Best) and Most Frequent (Vote) across five extraction fields.

gets are graded as wrong.[14]

$$\textit{Post-Fusion Accuracy} = \frac{\textit{\# People with Correct Target}}{\textit{\# People}}$$

Additionally, since the targets are ranked, we also calculated the mean reciprocal rank (MRR).[15] The data in Table 4 show the average system performance with the different fusion methods. Frequency voting gave anywhere from a 2% to a 20% improvement over picking the highest confidence candidate. CRF+E (the CRF trained with negative examples) was the highest performing system overall.

Birth Day		
	Fusion Accuracy	Fusion MRR
Rote Vote	.854	.877
NB+E Vote	.854	.889
CRF Vote	.650	.703
CRF+E Vote	**.883**	**.911**
Birth year		
Rote Vote	.387	.497
NB+E Vote	.778	.838
CRF Vote	.796	.860
CRF+E Vote	**.869**	**.876**
Occupation		
Rote Vote	.299	.405
NB+E Vote	**.642**	**.751**
CRF Vote	.606	.740
CRF+E Vote	.423	.553
Birthplace		
Rote Vote	.321	.338
NB+E Vote	**.474**	**.586**
CRF Vote	.321	.476
CRF+E Vote	.467	.560
Year of Death		
Rote Vote	.389	.389
NB+E Vote	.194	.383
CRF	**.750**	**.840**
CRF+E Vote	**.750**	.827

Table 5: Voting for information fusion, evaluated per person. CRF+E has best average performance (67.8%).

Table 5 shows the results of using each of these extractors to extract correct relationships from the top 150 ranked documents downloaded from the Web. CRF+E was a top performer in 3/5 of the cases. In the other 2 cases, the NB+E was the most successful, perhaps because NB+E's increased recall was more useful than CRF+E's improved precision.

[14]For year of death, we only graded cases where the person had died.

[15]The reciprocal rank = 1 / the rank of the correct target.

Retrieval Set Size and Performance

As with pre-fusion, we performed a set of experiments with different retrieval set sizes and used the CRF+E extraction system trained on 150 documents per person. The data in Figure 4 show that performance improves as the retrieval set size increases. Most of the gains come in the first 30 documents, where average performance increased from 14% (1 document) to 63% (30 documents). Increasing the retrieval set size to 150 documents per person yielded an additional 5% absolute improvement.

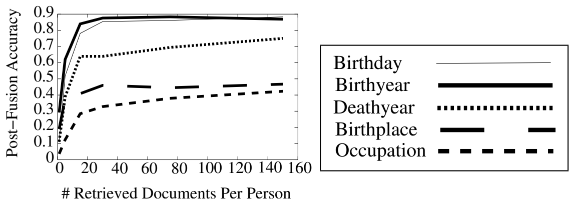


Figure 4: Fusion accuracy increases with more documents per person

Post-fusion errors come from two major sources. The first source is the misranking of correct relationships. The second is the case where relevant information is not retrieved at all, which we measure as

$$\textit{Post-Fusion Missing} = \frac{\textit{\# Missing Targets}}{\textit{\# People}}$$

The data in Figure 5 suggest that the decrease in missing targets is a significant contributing factor to the improvement in performance with increased document size. Missing targets were a major problem for Birthplace, constituting more than half the errors (32% at 150 documents).

4 Cross-Field Bootstrapping

Sections 2 and 3 presented methods for training separate extractors for particular relationships and for doing fusion across multiple documents. In this section, we leverage data interdependencies to improve performance.

The method we propose is to bootstrap across fields and use knowledge of one relationship to improve performance on the extraction of another. For

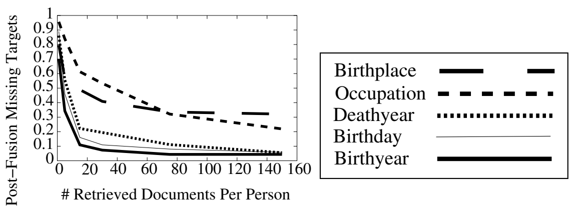


Figure 5: Additional documents decrease the number of post-fusion missing targets, targets which are never extracted in any document.

Birth year		
	Extraction Precision	Fusion Accuracy
CRF	.342	.797
+ birthday	.472	.861
CRF+E	.654	.869
+ birthday	**.809**	**.891**
Occupation		
	Extraction Precision	Fusion Accuracy
CRF	.219	.606
+ birthday	.217	.569
+ birth year(f)	21.9	.599
+ all	.214	.591
CRF+E	.246	.423
+ birthday	.325	.577
+ birth year(f)	**.387**	**.672**
+ all	.382	.642
Birthplace		
	Extraction Precision	Fusion Accuracy
CRF	.139	.321
+ birthday	.158	.372
+ birth year(f)	.156	.350
CRF+E	.357	.467
+ birthday	.350	.474
+ birth year(f)	.294	.350
+ occupation(f)	.314	.354
+ all	**.362**	**.532**

Table 6: Performance of Cross-Field Bootstrapping Models. (f) indicates that the best fused result was taken. birth year(f) means birth years were annotated using the system that discovered the most accurate birth years.

example, to extract birth year given knowledge of the birthday, in training we mark up each hook corpus D_x with the known birthday $b : birthday(x, b)$ and the target birth year $y : birthyear(x, y)$ and add an additional feature to the CRF that indicates whether the birthday has been seen in the sentence.[16] In testing, for each hook, we first find the birthday using the methods presented in the previous sections, annotate the corpus with the extracted birthday, and then apply the birth year CRF (see Figure 6 next page).

[16]The CRF state model doesn't change. When bootstrapping from multiple fields, we add the conjunctions of the fields as features.

Table 6 shows the effect of using this bootstrapped data to estimate other fields. Based on the relative performance of each of the individual extraction systems, we chose the following schedule for performing the bootstrapping: 1) Birthday, 2) Birth year, 3) Occupation, 4) Birthplace. We tried adding in all knowledge available to the system at each point in the schedule.[17] There are gains in accuracy for birth year, occupation and birthplace by using cross-field bootstrapping. The performance of the plain CRF+E averaged across all five fields is 67.4%, while for the best bootstrapped system it is 74.6%, a gain of 7%.

Doing bootstrapping in this way improves for people whose information is already partially correct. As a result, the percentage of people who have completely correct information improves to 37% from 13.8%, a gain of 24% over the non-bootstrapped CRF+E system. Additionally, erroneous extractions do not hurt accuracy on extraction of other fields. Performance in the bootstrapped system for birthyear, occupation and birth place when the birthday is wrong is almost the same as performance in the non-bootstrapped system.

5 Training Set Size Reduction

One of the results from Section 2 is that lower ranked documents are less likely to contain the relevant biographic information. While this does not have an dramatic effect on the post-fusion accuracy (which improves with more documents), it suggests that training on a smaller corpus, with more relevant documents and more sentences with the desired information, might lead to equivalent or improved performance. In a final set of experiments we looked at system performance when the extractor is trained on fewer than 150 documents per person.

The data in Figure 7 show that training on 30 documents per person yields around the same performance as training on 150 documents per person. Average performance when the system was trained on 30 documents per person is 70%, while average performance when trained on 150 documents per person is 68%. Most of this loss in performance comes from losses in occupation, but the other relationships

[17]This system has the extra knowledge of which fused method is the best for each relationship. This was assessed by inspection.

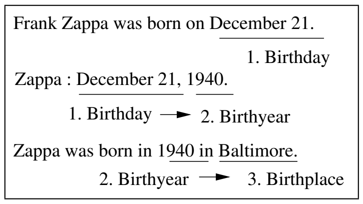

Frank Zappa was born on December 21.
1. Birthday
Zappa : December 21, 1940.
1. Birthday → 2. Birthyear
Zappa was born in 1940 in Baltimore.
2. Birthyear → 3. Birthplace

Figure 6: Cross-Field Bootstrapping: In step (1) The birthday, December 21, is extracted and the text marked. In step 2, cooccurrences with the discovered birthday make 1940 a better candidate for birthyear. In step (3), the discovered birthyear appears in contexts where the discovered birthday does not and improves extraction of birth place.

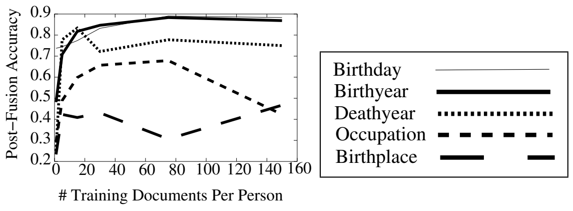


Figure 7: Fusion accuracy doesn't improve with more than 30 training documents per person.

have either little or no gain from training on additional documents. There are two possible reasons why more training data may not help, and even may hurt performance.

One possibility is that higher ranked retrieved documents are more likely to contain biographical facts, while in later documents it is more likely that automatically annotated training instances are in fact false positives. That is, higher ranked documents are cleaner training data. Pre-Fusion precision results (Figure 8) support this hypothesis since it appears that later instances are often contaminating earlier models.

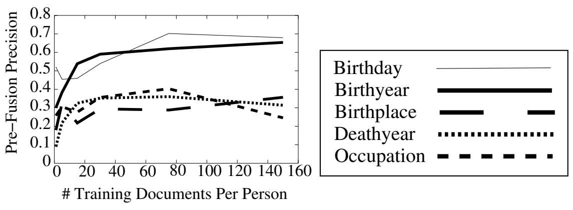


Figure 8: Pre-Fusion precision shows slight drops with increased training documents.

The data in Figure 9 suggest an alternate possibility that later documents also shift the prior toward a model where it is less likely that a relationship is observed as fewer targets are extracted.

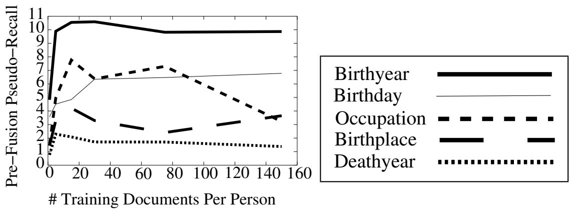


Figure 9: Pre-Fusion Pseudo-Recall also drops with increased training documents.

6 Related Work

The closest related work to the task of biographic fact extraction was done by Cowie et al. (2000) and Schiffman et al. (2001), who explore the problem of biographic summarization.

There has been rather limited published work in multi-document information extraction. The closest work to what we present here is Masterson and Kushmerick (2003), who perform multi-document information extraction trained on manually annotated training data and use Best Confidence to resolve each particular template slot. In summarizarion, many systems have examined the multi-document case. Notable systems are SUMMONS (Radev and McKeown, 1998) and RIPTIDE (White et al., 2001), which assume perfect extracted information and then perform closed domain summarization. Barzilay et al. (1999) does not explicitly extract facts, but instead picks out relevant repeated elements and combines them to obtain a summary which retains the semantics of the original.

In recent question answering research, information fusion has been used to combine multiple candidate answers to form a consensus answer. Clarke et al. (2001) use frequency of n-gram occurrence to pick answers for particular questions. Another example of answer fusion comes in (Brill et al., 2001) which combines the output of multiple question answering systems in order to rank answers. Dalmas and Webber (2004) use a WordNet cover heuristic to choose an appropriate location from a large candidate set of answers.

There has been a considerable amount of work in training information extraction systems from annotated data since the mid-90s. The initial work in the field used lexico-syntactic template patterns learned using a variety of different empirical approaches (Riloff and Schmelzenbach, 1998; Huffman, 1995;

Soderland et al., 1995). Seymore et al. (1999) use HMMs for information extraction and explore ways to improve the learning process.

Nahm and Mooney (2002) suggest a method to learn word-to-word relationships across fields by doing data mining on information extraction results. Prager et al. (2004) uses knowledge of birth year to weed out candidate years of death that are impossible. Using the CRF extractors in our data set, this heuristic did not yield any improvement. More distantly related work for multi-field extraction suggests methods for combining information in graphical models across multiple extraction instances (Sutton et al., 2004; Bunescu and Mooney, 2004) .

7 Conclusion

This paper has presented new experimental methodologies and results for cross-document information fusion, focusing on the task of biographic fact extraction and has proposed a new method for cross-field bootstrapping. In particular, we have shown that automatic annotation can be used effectively to train statistical information extractors such Naïve Bayes and CRFs, and that CRF extraction accuracy can be improved by 5% with a negative example model. We looked at cross-document fusion and demonstrated that voting outperforms choosing the highest confidence extracted information by 2% to 20%. Finally, we introduced a cross-field bootstrapping method that improved average accuracy by 7%.

References

E. Agichtein and L. Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of ICDL*, pages 85–94.

R. Barzilay, K. R. McKeown, and M. Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of ACL*, pages 550–557.

E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-intensive question answering. In *Proceedings of TREC*, pages 183–189.

S. Brin. 1998. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*, pages 172–183.

R. Bunescu and R. Mooney. 2004. Collective information extraction with relational markov networks. In *Proceedings of ACL*, pages 438–445.

C. L. A. Clarke, G. V. Cormack, and T. R. Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of SIGIR*, pages 358–365.

J. Cowie, S. Nirenburg, and H. Molina-Salgado. 2000. Generating personal profiles. In *The International Conference On MT And Multilingual NLP*.

T. Dalmas and B. Webber. 2004. Information fusion for answering factoid questions. In *Proceedings of 2nd CoLogNET-ElsNET Symposium. Questions and Answers: Theoretical Perspectives*.

D. Freitag and A. McCallum. 1999. Information extraction with hmms and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 31–36.

S. B. Huffman. 1995. Learning information extraction patterns from examples. In *Working Notes of the IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing*, pages 127–134.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

T. R. Leek. 1997. Information extraction using hidden markov models. Master's Thesis, UC San Diego.

D. Masterson and N. Kushmerick. 2003. Information extraction from multi-document threads. In *Proceedings of ECML-2003: Workshop on Adaptive Text Extraction and Mining*, pages 34–41.

A. McCallum. 2002. Mallet: A machine learning for language toolkit.

U. Nahm and R. Mooney. 2002. Text mining with information extraction. In *Proceedings of the AAAI 2220 Spring Symposium on Mining Answers from Texts and Knowledge Bases*, pages 60–67.

J. Prager, J. Chu-Carroll, and K. Czuba. 2004. Question answering by constraint satisfaction: Qa-by-dossier with constraints. In *Proceedings of ACL*, pages 574–581.

D. R. Radev and K. R. McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500.

D. Ravichandran and E. Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL*, pages 41–47.

E. Riloff and M. Schmelzenbach. 1998. An empirical approach to conceptual case frame acquisition. In *Proceedings of WVLC*, pages 49–56.

E. Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of AAAI*, pages 1044–1049.

B. Schiffman, I. Mani, and K. J. Concepcion. 2001. Producing biographical summaries: Combining linguistic knowledge with corpus statistics. In *Proceedings of ACL*, pages 450–457.

K. Seymore, A. McCallum, and R. Rosenfeld. 1999. Learning hidden markov model structure for information extraction. In *AAAI'99 Workshop on Machine Learning for Information Extraction*, pages 37–42.

S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. 1995. CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of IJCAI*, pages 1314–1319.

C. Sutton, K. Rohanimanesh, and A. McCallum. 2004. Dynamic conditional random fields: factorize probabilistic models for labeling and segmenting sequence data. In *Proceedings of ICML*.

M. White, T. Korelsky, C. Cardie, V. Ng, D. Pierce, and K. Wagstaff. 2001. Multi-document summarization via information extraction. In *Proceedings of HLT*.

Simple Algorithms for Complex Relation Extraction with Applications to Biomedical IE

Ryan McDonald[1] **Fernando Pereira**[1] **Seth Kulick**[2]
[1]CIS and [2]IRCS, University of Pennsylvania, Philadelphia, PA
{ryantm,pereira}@cis.upenn.edu, skulick@linc.cis.upenn.edu

Scott Winters **Yang Jin** **Pete White**
Division of Oncology, Children's Hospital of Pennsylvania, Philadelphia, PA
{winters,jin,white}@genome.chop.edu

Abstract

A complex relation is any n-ary relation in which some of the arguments may be be unspecified. We present here a simple two-stage method for extracting complex relations between named entities in text. The first stage creates a graph from pairs of entities that are likely to be related, and the second stage scores maximal cliques in that graph as potential complex relation instances. We evaluate the new method against a standard baseline for extracting genomic variation relations from biomedical text.

1 Introduction

Most research on text information extraction (IE) has focused on accurate tagging of named entities. Successful early named-entity taggers were based on finite-state generative models (Bikel et al., 1999). More recently, discriminatively-trained models have been shown to be more accurate than generative models (McCallum et al., 2000; Lafferty et al., 2001; Kudo and Matsumoto, 2001). Both kinds of models have been developed for tagging entities such as people, places and organizations in news material. However, the rapid development of bioinformatics has recently generated interest on the extraction of biological entities such as genes (Collier et al., 2000) and genomic variations (McDonald et al., 2004b) from biomedical literature.

The next logical step for IE is to begin to develop methods for extracting meaningful relations involving named entities. Such relations would be extremely useful in applications like question answering, automatic database generation, and intelligent document searching and indexing. Though not as well studied as entity extraction, relation extraction has still seen a significant amount of work. We discuss some previous approaches at greater length in Section 2.

Most relation extraction systems focus on the specific problem of extracting binary relations, such as the *employee of* relation or *protein-protein interaction* relation. Very little work has been done in recognizing and extracting more complex relations. We define a *complex relation* as any n-ary relation among n typed entities. The relation is defined by the *schema* $(t_1, \ldots, t_n)$ where $t_i \in T$ are entity types. An *instance* (or *tuple*) in the relation is a list of entities $(e_1, \ldots, e_n)$ such that either type$(e_i) = t_i$, or $e_i = \perp$ indicating that the ith element of the tuple is missing.

For example, assume that the entity types are $T = \{\mathsf{person}, \mathsf{job}, \mathsf{company}\}$ and we are interested in the ternary relation with schema $(\mathsf{person}, \mathsf{job}, \mathsf{company})$ that relates a person to their job at a particular company. For the sentence *"John Smith is the CEO at Inc. Corp."*, the system would ideally extract the tuple $(\textit{John Smith}, \textit{CEO}, \textit{Inc. Corp.})$. However, for the sentence *"Everyday John Smith goes to his office at Inc. Corp."*, the system would extract $(\textit{John Smith}, \perp, \textit{Inc. Corp.})$, since there is no mention of a job title. Hence, the goal of complex relation extraction is to identify all instances of the relation of interest in some piece of text, including

Proceedings of the 43rd Annual Meeting of the ACL, pages 491–498,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

incomplete instances.

We present here several simple methods for extracting complex relations. All the methods start by recognized pairs of entity mentions, that is, binary relation instances, that appear to be arguments of the relation of interest. Those pairs can be seen as the edges of a graph with entity mentions as nodes. The algorithms then try to reconstruct complex relations by making tuples from selected maximal cliques in the graph. The methods are general and can be applied to any complex relation fitting the above definition. We also assume throughout the paper that the entities and their type are known a priori in the text. This is a fair assumption given the current high standard of state-of-the-art named-entity extractors.

A primary advantage of factoring complex relations into binary relations is that it allows the use of standard classification algorithms to decide whether particular pairs of entity mentions are related. In addition, the factoring makes training data less sparse and reduces the computational cost of extraction. We will discuss these benefits further in Section 4.

We evaluated the methods on a large set of annotated biomedical documents to extract relations related to genomic variations, demonstrating a considerable improvement over a reasonable baseline.

2 Previous work

A representative approach to relation extraction is the system of Zelenko et al. (2003), which attempts to identify binary relations in news text. In that system, each pair of entity mentions of the correct types in a sentence is classified as to whether it is a positive instance of the relation. Consider the binary relation *employee of* and the sentence *"John Smith, not Jane Smith, works at IBM"*. The pair $(\textit{John Smith}, \textit{IBM})$ is a positive instance, while the pair $(\textit{Jane Smith}, \textit{IBM})$ is a negative instance. Instances are represented by a pair of entities and their position in a shallow parse tree for the containing sentence. Classification is done by a support-vector classifier with a specialized kernel for that shallow parse representation.

This approach — enumerating all possible entity pairs and classifying each as positive or negative — is the standard method in relation extraction. The main differences among systems are the choice of trainable classifier and the representation for instances.

For binary relations, this approach is quite tractable: if the relation schema is (t_1, t_2), the number of potential instances is $O(|t_1|\,|t_2|)$, where $|t|$ is the number of entity mentions of type t in the text under consideration.

One interesting system that does not belong to the above class is that of Miller et al. (2000), who take the view that relation extraction is just a form of probabilistic parsing where parse trees are augmented to identify all relations. Once this augmentation is made, any standard parser can be trained and then run on new sentences to extract new relations. Miller et al. show such an approach can yield good results. However, it can be argued that this method will encounter problems when considering anything but binary relations. Complex relations would require a large amount of tree augmentation and most likely result in extremely sparse probability estimates. Furthermore, by integrating relation extraction with parsing, the system cannot consider long-range dependencies due to the local parsing constraints of current probabilistic parsers. The higher the arity of a relation, the more likely it is that entities will be spread out within a piece of text, making long range dependencies especially important.

Roth and Yih (2004) present a model in which entity types and relations are classified jointly using a set of global constraints over locally trained classifiers. This joint classification is shown to improve accuracy of both the entities and relations returned by the system. However, the system is based on constraints for binary relations only.

Recently, there has also been many results from the biomedical IE community. Rosario and Hearst (2004) compare both generative and discriminative models for extracting seven relationships between treatments and diseases. Though their models are very flexible, they assume at most one relation per sentence, ruling out cases where entities participate in multiple relations, which is a common occurrence in our data. McDonald et al. (2004a) use a rule-based parser combined with a rule-based relation identifier to extract generic binary relations between biological entities. As in predicate-argument extraction (Gildea and Jurafsky, 2002), each relation is always associated with a verb in the sentence that specifies the relation type. Though this system is very general, it is limited by the fact that the design ignores relations not expressed by a verb, as the *employee of* relation in *"John Smith, CEO of Inc. Corp., announced he will resign"*.

Most relation extraction systems work primarily on a sentential level and never consider relations that cross sentences or paragraphs. Since current data sets typically only annotate intra-sentence relations, this has not yet proven to be a problem.

3 Definitions

3.1 Complex Relations

Recall that a complex n-ary relation is specified by a schema $(t_1, \dots, t_n)$ where $t_i \in T$ are entity types. Instances of the relation are tuples $(e_1, \dots, e_n)$ where either type$(e_i) = t_i$, or $e_i = \perp$ (missing argument). The only restriction this definition places on a relation is that the arity must be known. As we discuss it further in Section 6, this is not required by our methods but is assumed here for simplicity. We also assume that the system works on a single relation type at a time, although the methods described here are easily generalizable to systems that can extract many relations at once.

3.2 Graphs and Cliques

An *undirected graph* $G = (V, E)$ is specified by a set of *vertices* V and a set of *edges* E, with each edge an unordered pair (u, v) of vertices. $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' = \{(u, v) : u, v \in V', (u, v) \in E\}$. A *clique* C of G is a subgraph of G in which there is an edge between every pair of vertices. A *maximal clique* of G is a clique $C = (V_C, E_C)$ such that there is no other clique $C' = (V_{C'}, E_{C'})$ such that $V_C \subset V_{C'}$.

4 Methods

We describe now a simple method for extracting complex relations. This method works by first factoring all complex relations into a set of binary relations. A classifier is then trained in the standard manner to recognize all pairs of related entities. Finally a graph is constructed from the output of this classifier and the complex relations are determined from the cliques of this graph.

a. All possible relation instances

(*John*, *CEO*, *Inc. Corp.*)
(*John*, $\perp$, *Inc. Corp.*)
(*John*, *CEO*, *Biz. Corp.*)
(*John*, $\perp$, *Biz. Corp.*)
(*John*, *CEO*, $\perp$)
(*Jane*, *CEO*, *Inc. Corp.*)
(*Jane*, $\perp$, *Inc. Corp.*)
(*Jane*, *CEO*, *Biz. Corp.*)
(*Jane*, $\perp$, *Biz. Corp.*)
(*Jane*, *CEO*, $\perp$)
($\perp$, *CEO*, *Inc. Corp.*)
($\perp$, *CEO*, *Biz. Corp.*)

b. All possible binary relations

(*John*, *CEO*)
(*John*, *Inc. Corp.*)
(*John*, *Biz. Corp.*)
(*CEO*, *Inc. Corp.*)
(*CEO*, *Biz. Corp.*)
(*Jane*, *CEO*)
(*Jane*, *Inc. Corp.*)
(*Jane*, *Biz. Corp.*)

Figure 1: Relation factorization of the sentence: *John and Jane are CEOs at Inc. Corp. and Biz. Corp. respectively.*

4.1 Classifying Binary Relations

Consider again the motivating example of the (person, job, company) relation and the sentence *"John and Jane are CEOs at Inc. Corp. and Biz. Corp. respectively"*. This sentence contains two people, one job title and two companies.

One possible method for extracting the relation of interest would be to first consider all 12 possible tuples shown in Figure 1a. Using all these tuples, it should then be possible to train a classifier to distinguish valid instances such as (*John*, *CEO*, *Inc. Corp.*) from invalid ones such as (*Jane*, *CEO*, *Inc. Corp.*). This is analogous to the approach taken by Zelenko et al. (2003) for binary relations.

There are problems with this approach. Computationally, for an n-ary relation, the number of possible instances is $O(|t_1|\,|t_2| \cdots |t_n|)$. Conservatively, letting m be the smallest $|t_i|$, the run time is $O(m^n)$, exponential in the arity of the relation. The second problem is how to manage incomplete but correct instances such as (*John*, $\perp$, *Inc. Corp.*) when training the classifier. If this instance is marked as negative, then the model might incorrectly disfavor features that correlate *John* to *Inc. Corp.*. However, if this instance is labeled positive, then the model may tend to prefer the shorter and more compact incomplete relations since they will be abundant in the positive training examples. We could always ignore instances of this form, but then the data would be heavily skewed towards negative instances.

Instead of trying to classify all possible relation instances, in this work we first classify pairs of entities as being related or not. Then, as discussed in Section 4.2, we reconstruct the larger complex relations from a set of binary relation instances.

Factoring relations into a set of binary decisions has several advantages. The set of possible pairs is much smaller then the set of all possible complex relation instances. This can be seen in Figure 1b, which only considers pairs that are consistent with the relation definition. More generally, the number of pairs to classify is $O((\sum_i |t_i|)^2)$, which is far better than the exponentially many full relation instances. There is also no ambiguity when labeling pairs as positive or negative when constructing the training data. Finally, we can rely on previous work on classification for binary relation extraction to identify pairs of related entities.

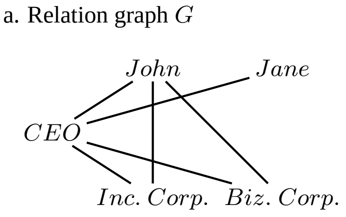
To train a classifier to identify pairs of related entities, we must first create the set of all positive and negative pairs in the data. The positive instances are all pairs that occur together in a valid tuple. For the example sentence in Figure 1, these include the pairs (*John*, *CEO*), (*John*, *Inc. Corp.*), (*CEO*, *Inc. Corp.*), (*CEO*, *Biz. Corp.*), (*Jane*, *CEO*) and (*Jane*, *Biz. Corp.*). To gather negative instances, we extract all pairs that never occur together in a valid relation. From the same example these would be the pairs (*John*, *Biz. Corp.*) and (*Jane*, *Inc. Corp.*).

This leads to a large set of positive and negative binary relation instances. At this point we could employ any binary relation classifier and learn to identify new instances of related pairs of entities. We use a standard maximum entropy classifier (Berger et al., 1996) implemented as part of MALLET (McCallum, 2002). The model is trained using the features listed in Table 1.

This is a very simple binary classification model. No deep syntactic structure such as parse trees is used. All features are basically over the words separating two entities and their part-of-speech tags. Of course, it would be possible to use more syntactic information if available in a manner similar to that of Zelenko et al. (2003). However, the primary purpose of our experiments was not to create a better binary relation extractor, but to see if complex relations could be extracted through binary factorization followed by reconstruction. In Section 5.2 we present an empirical evaluation of the binary relation classifier.

Feature Set
entity type of e_1 and e_2
words in e_1 and e_2
word bigrams in e_1 and e_2
POS of e_1 and e_2
words between e_1 and e_2
word bigrams between e_1 and e_2
POS between e_1 and e_2
distance between e_1 and e_2
concatenations of above features

Table 1: Feature set for maximum entropy binary relation classifier. e_1 and e_2 are entities.



b. Tuples from G
(*John*, *CEO*, $\perp$)
(*John*, $\perp$, *Inc. Corp.*)
(*John*, $\perp$, *Biz. Corp.*)
(*Jane*, *CEO*, $\perp$)
($\perp$, *CEO*, *Inc. Corp.*)
($\perp$, *CEO*, *Biz. Corp.*)
(*John*, *CEO*, *Inc. Corp.*)
(*John*, *CEO*, *Biz. Corp.*)

Figure 2: Example of a relation graph and tuples from all the cliques in the graph.

4.2 Reconstructing Complex Relations

4.2.1 Maximal Cliques

Having identified all pairs of related entities in the text, the next stage is to reconstruct the complex relations from these pairs. Let $G = (V, E)$ be an undirected graph where the vertices V are entity mentions in the text and the edges E represent binary relations between entities. We reconstruct the complex relation instances by finding maximal cliques in the graphs.

The simplest approach is to create the graph so that two entities in the graph have an edge if the binary classifier believes they are related. For example, consider the binary factorization in Figure 1 and imagine the classifier identified the following pairs as being related: (*John*, *CEO*), (*John*, *Inc. Corp.*), (*John*, *Biz. Corp.*), (*CEO*, *Inc. Corp.*), (*CEO*, *Biz. Corp.*) and (*Jane*, *CEO*). The resulting graph can be seen in Figure 2a.

Looking at this graph, one solution to constructing complex relations would be to consider all the cliques in the graph that are consistent with the definition of the relation. This is equivalent to having the system return only relations in which the binary classifier believes that all of the entities involved are pairwise related. All the cliques in the example are shown in Figure 2b. We add $\perp$ fields to the tuples to be consistent with the relation definition.

This could lead to a set of overlapping cliques, for instance $(\textit{John}, \textit{CEO}, \textit{Inc. Corp.})$ and $(\textit{John}, \textit{CEO}, \perp)$. Instead of having the system return all cliques, our system just returns the maximal cliques, that is, those cliques that are not subsets of other cliques. Hence, for the example under consideration in Figure 2, the system would return the one correct relation, $(\textit{John}, \textit{CEO}, \textit{Inc. Corp.})$, and two incorrect relations, $(\textit{John}, \textit{CEO}, \textit{Biz. Corp.})$ and $(\textit{Jane}, \textit{CEO}, \perp)$. The second is incorrect since it does not specify the *company* slot of the relation even though that information is present in the text.

It is possible to find degenerate sentences in which perfect binary classification followed by maximal clique reconstruction will lead to errors. One such sentence is, *"John is C.E.O. and C.F.O. of Inc. Corp. and Biz. Corp. respectively and Jane vice-versa"*. However, we expect such sentences to be rare; in fact, they never occur in our data.

The real problem with this approach is that an arbitrary graph can have exponentially many cliques, negating any efficiency advantage over enumerating all n-tuples of entities. Fortunately, there are algorithms for finding all maximal cliques that are efficient in practice. We use the algorithm of Bron and Kerbosch (1973). This is a well known branch and bound algorithm that has been shown to empirically run linearly in the number of maximal cliques in the graph. In our experiments, this algorithm found all maximal cliques in a matter of seconds.

4.2.2 Probabilistic Cliques

The above approach has a major shortcoming in that it assumes the output of the binary classifier to be absolutely correct. For instance, the classifier may have thought with probability 0.49, 0.99 and 0.99 that the following pairs were related: $(\textit{Jane}, \textit{Biz. Corp.})$, $(\textit{CEO}, \textit{Biz. Corp.})$ and $(\textit{Jane}, \textit{CEO})$ respectively. The maximal clique method would not produce the tuple $(\textit{Jane}, \textit{CEO}, \textit{Biz. Corp.})$ since it never considers the edge between *Jane* and *Biz. Corp.* However, given the probability of the edges, we would almost certainly want this tuple returned.

What we would really like to model is a belief that *on average* a clique represents a valid relation instance. To do this we use the complete graph $G = (V, E)$ with edges between all pairs of entity mentions. We then assign weight $w(e)$ to edge e equal to the probability that the two entities in e are related, according to the classifier. We define the weight of a clique $w(C)$ as the mean weight of the edges in the clique. Since edge weights represent probabilities (or ratios), we use the geometric mean

$$w(C) = \left(\prod_{e \in E_C} w(e) \right)^{1/|E_C|}$$

We decide that a clique C represents a valid tuple if $w(C) \geq 0.5$. Hence, the system finds all maximal cliques as before, but considers only those where $w(C) \geq 0.5$, and it may select a non-maximal clique if the weight of all larger cliques falls below the threshold. The cutoff of 0.5 is not arbitrary, since it ensures that the average probability of a clique representing a relation instance is at least as large as the average probability of it not representing a relation instance. We ran experiments with varying levels of this threshold and found that, roughly, lower thresholds result in higher precision at the expense of recall since the system returns fewer but larger tuples. Optimum results were obtained for a cutoff of approximately 0.4, but we report results only for $w(C) \geq 0.5$.

The major problem with this approach is that there will always be exponentially many cliques since the graph is fully connected. However, in our experiments we pruned all edges that would force any containing clique C to have $w(C) < 0.5$. This typically made the graphs very sparse.

Another problem with this approach is the assumption that the binary relation classifier outputs probabilities. For maximum entropy and other probabilistic frameworks this is not an issue. However, many classifiers, such as SVMs, output scores or distances. It is possible to transform the scores from those models through a sigmoid to yield probabilities, but there is no guarantee that those probability values will be well calibrated.

5 Experiments

5.1 Problem Description and Data

We test these methods on the task of extracting genomic variation events from biomedical text (McDonald et al., 2004b). Briefly, we define a variation event as an acquired genomic aberration: a specific, one-time alteration at the genomic level and described at the nucleic acid level, amino acid level or both. Each variation event is identified by the relationship between a *type* of variation, its *location*, and the corresponding state change from an *initial-state* to an *altered-state*. This can be formalized as the following complex schema

$$(\textsf{var-type}, \textsf{location}, \textsf{initial-state}, \textsf{altered-state})$$

A simple example is the sentence

"At codons 12 and 61, the occurrence of point mutations from G/A to T/G were observed"

which gives rise to the tuples

$(\textit{point mutation, codon 12, G, T})$
$(\textit{point mutation, codon 61, A, G})$

Our data set consists of 447 abstracts selected from MEDLINE as being relevant to populating a database with facts of the form: *gene X with variation event Y is associated with malignancy Z*. Abstracts were randomly chosen from a larger corpus identified as containing variation mentions pertaining to cancer.

The current data consists of 4691 sentences that have been annotated with 4773 entities and 1218 relations. Of the 1218 relations, 760 have two $\perp$ arguments, 283 have one $\perp$ argument, and 175 have no $\perp$ arguments. Thus, 38% of the relations tagged in this data cannot be handled using binary relation classification alone. In addition, 4% of the relations annotated in this data are non-sentential. Our system currently only produces sentential relations and is therefore bounded by a maximum recall of 96%. Finally, we use gold standard entities in our experiments. This way we can evaluate the performance of the relation extraction system isolated from any kind of pipelined entity extraction errors. Entities in this domain can be found with fairly high accuracy (McDonald et al., 2004b).

It is important to note that just the presence of two entity types does not entail a relation between them. In fact, 56% of entity pairs are not related, due either to explicit disqualification in the text (e.g. "... *the lack of G to T transversion ...*") or ambiguities that arise from multiple entities of the same type.

5.2 Results

Because the data contains only 1218 examples of relations we performed 10-fold cross-validation tests for all results. We compared three systems:

- **MC:** Uses the maximum entropy binary classifier coupled with the maximal clique complex relation reconstructor.
- **PC:** Same as above, except it uses the probabilistic clique complex relation reconstructor.
- **NE:** A maximum entropy classifier that naively enumerates all possible relation instances as described in Section 4.1.

In training system **NE**, all incomplete but correct instances were marked as positive since we found this had the best performance. We used the same pairwise entity features in the binary classifier of the above two systems. However, we also added higher order versions of the pairwise features. For this system we only take maximal relations,that is, if $(\textit{John}, \textit{CEO}, \textit{Inc. Corp.})$ and $(\textit{John}, \perp, \textit{Inc. Corp.})$ are both labeled positive, the system would only return the former.

Table 2 contains the results of the maximum entropy binary relation classifier (used in systems **MC** and **PC**). The 1218 annotated complex relations produced 2577 unique binary pairs of related entities. We can see that the maximum entropy classifier performs reasonably well, although performance may be affected by the lack of rich syntactic features, which have been shown to help performance (Miller et al., 2000; Zelenko et al., 2003).

Table 3 compares the three systems on the real problem of extracting complex relations. An extracted complex relation is considered correct if and only if all the entities in the relation are correct. There is no partial credit. All training and clique finding algorithms took under 5 minutes for the entire data set. Naive enumeration took approximately 26 minutes to train.

ACT	PRD	COR
2577	2722	2101
Prec	**Rec**	**F-Meas**
0.7719	0.8153	0.7930

Table 2: Binary relation classification results for the maximum entropy classifier. ACT: actual number of related pairs, PRD: predicted number of related pairs and COR: correctly identified related pairs.

System	Prec	Rec	F-Meas
NE	0.4588	0.6995	0.5541
MC	0.5812	0.7315	0.6480
PC	0.6303	0.7726	0.6942

Table 3: Full relation classification results. For a relation to be classified correctly, all the entities in the relation must be correctly identified.

First we observe that the maximal clique method combined with maximum entropy (system **MC**) reduces the relative error rate over naively enumerating and classifying all instances (system **NE**) by 21%. This result is very positive. The system based on binary factorization not only is more efficient then naively enumerating all instances, but significantly outperforms it as well. The main reason naive enumeration does so poorly is that all correct but incomplete instances are marked as positive. Thus, even slight correlations between partially correct entities would be enough to classify an instance as correct, which results in relatively good recall but poor precision. We tried training only with correct and complete positive instances, but the result was a system that only returned few relations since negative instances overwhelmed the training set. With further tuning, it may be possible to improve the performance of this system. However, we use it only as a baseline and to demonstrate that binary factorization is a feasible and accurate method for extracting complex relations.

Furthermore, we see that using probabilistic cliques (system **PC**) provides another large improvement, a relative error reduction of 13% over using maximal cliques and 31% reduction over enumeration. Table 4 shows the breakdown of relations returned by type. There are three types of relations, 2-ary, 3-ary and 4-ary, each with 2, 1 and 0 $\perp$ arguments respectively, e.g.

System	2-ary	3-ary	4-ary
NE	760:1097:600	283:619:192	175:141:60
MC	760:1025:601	283:412:206	175:95:84
PC	760:870:590	283:429:223	175:194:128

Table 4: Breakdown of true positive relations by type that were returned by each system. Each cell contains three numbers, Actual:Predicted:Correct, which represents for each arity the actual, predicted and correct number of relations for each system.

$(\textit{point mutation}, \textit{codon 12}, \perp, \perp)$ is a 2-ary relation. Clearly the probabilistic clique method is much more likely to find larger non-binary relations, verifying the motivation that there are some low probability edges that can still contribute to larger cliques.

6 Conclusions and Future Work

We presented a method for complex relation extraction, the core of which was to factorize complex relations into sets of binary relations, learn to identify binary relations and then reconstruct the complex relations by finding maximal cliques in graphs that represent relations between pairs of entities. The primary advantage of this method is that it allows for the use of almost any binary relation classifier, which have been well studied and are often accurate. We showed that such a method can be successful with an empirical evaluation on a large set of biomedical data annotated with genomic variation relations. In fact, this approach is both significantly quicker and more accurate then enumerating and classifying all possible instances. We believe this work provides a good starting point for continued research in this area.

A distinction may be made between the factored system presented here and one that attempts to classify complex relations without factorization. This is related to the distinction between methods that learn local classifiers that are combined with global constraints after training and methods that incorporate the global constraints into the learning process. McCallum and Wellner (2003) showed that learning binary co-reference relations globally improves performance over learning relations in isolation. However, their model relied on the transitive property inherent in the co-reference relation. Our system can be seen as an instance of a local learner. Punyakanok et al. (2004) argued that local learning actually outperforms global learning in cases when local decisions can easily be learnt by the classifier. Hence, it is reasonable to assume that our binary factorization method will perform well when binary relations can be learnt with high accuracy.

As for future work, there are many things that we plan to look at. The binary relation classifier we employ is quite simplistic and most likely can be improved by using features over a deeper representation of the data such as parse trees. Other more powerful binary classifiers should be tried such as those based on tree kernels (Zelenko et al., 2003). We also plan on running these algorithms on more data sets to test if the algorithms empirically generalize to different domains.

Perhaps the most interesting open problem is how to learn the complex reconstruction phase. One possibility is recent work on supervised clustering. Letting the edge probabilities in the graphs represent a distance in some space, it may be possible to learn how to cluster vertices into relational groups. However, since a vertex/entity can participate in one or more relation, any clustering algorithm would be required to produce non-disjoint clusters.

We mentioned earlier that the only restriction of our complex relation definition is that the arity of the relation must be known in advance. It turns out that the algorithms we described can actually handle dynamic arity relations. All that is required is to remove the constraint that maximal cliques must be consistent with the structure of the relation. This represents another advantage of binary factorization over enumeration, since it would be infeasible to enumerate all possible instances for dynamic arity relations.

Acknowledgments
The authors would like to thank Mark Liberman, Mark Mandel and Eric Pancoast for useful discussion, suggestions and technical support. This work was supported in part by NSF grant ITR 0205448.

References

A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1).

D.M. Bikel, R. Schwartz, and R.M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning Journal Special Issue on Natural Language Learning*, 34(1/3):221–231.

C. Bron and J. Kerbosch. 1973. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577.

N. Collier, C. Nobata, and J. Tsujii. 2000. Extracting the names of genes and gene products with a hidden Markov model. In *Proc. COLING*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proc. NAACL*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Information Integration on the Web*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. ICML*.

A. K. McCallum. 2002. MALLET: A machine learning for language toolkit.

D.M. McDonald, H. Chen, H. Su, and B.B. Marshall. 2004a. Extracting gene pathway relations using a hybrid grammar: the Arizona Relation Parser. *Bioinformatics*, 20(18):3370–78.

R.T. McDonald, R.S. Winters, M. Mandel, Y. Jin, P.S. White, and F. Pereira. 2004b. An entity tagger for recognizing acquired genomic variations in cancer literature. *Bioinformatics*, 20(17):3249–3251.

S. Miller, H. Fox, L.A. Ramshaw, and R.M. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proc. NAACL*.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Learning via inference over structurally constrained output. In *Workshop on Learning Structured with Output, NIPS*.

Barbara Rosario and Marti A. Hearst. 2004. Classifying semantic relations in bioscience texts. In *ACL*.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. CoNLL*.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *JMLR*.

Resume Information Extraction with Cascaded Hybrid Model

Kun Yu
Department of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui, China, 230027
yukun@mail.ustc.edu.cn

Gang Guan
Department of Electronic Engineering
Tsinghua University
Bejing, China, 100084
guangang@tsinghua.org.cn

Ming Zhou
Microsoft Research Asia
5F Sigma Center, No.49 Zhichun Road, Haidian
Bejing, China, 100080
mingzhou@microsoft.com

Abstract

This paper presents an effective approach for resume information extraction to support automatic resume management and routing. A cascaded information extraction (IE) framework is designed. In the first pass, a resume is segmented into a consecutive blocks attached with labels indicating the information types. Then in the second pass, the detailed information, such as *Name* and *Address*, are identified in certain blocks (e.g. blocks labelled with *Personal Information*), instead of searching globally in the entire resume. The most appropriate model is selected through experiments for each IE task in different passes. The experimental results show that this cascaded hybrid model achieves better F-score than flat models that do not apply the hierarchical structure of resumes. It also shows that applying different IE models in different passes according to the contextual structure is effective.

1 Introduction

Big enterprises and head-hunters receive hundreds of resumes from job applicants every day. Automatically extracting structured information from resumes of different styles and formats is needed to support the automatic construction of database, searching and resume routing. The definition of resume information fields varies in different applications. Normally, resume information is described as a hierarchical structure with two layers. The first layer is composed of consecutive general information blocks such as *Personal Information*, *Education* etc. Then within each general information block, detailed information pieces can be found, e.g., in *Personal Information* block, detailed information such as *Name*, *Address*, *Email* etc. can be further extracted.

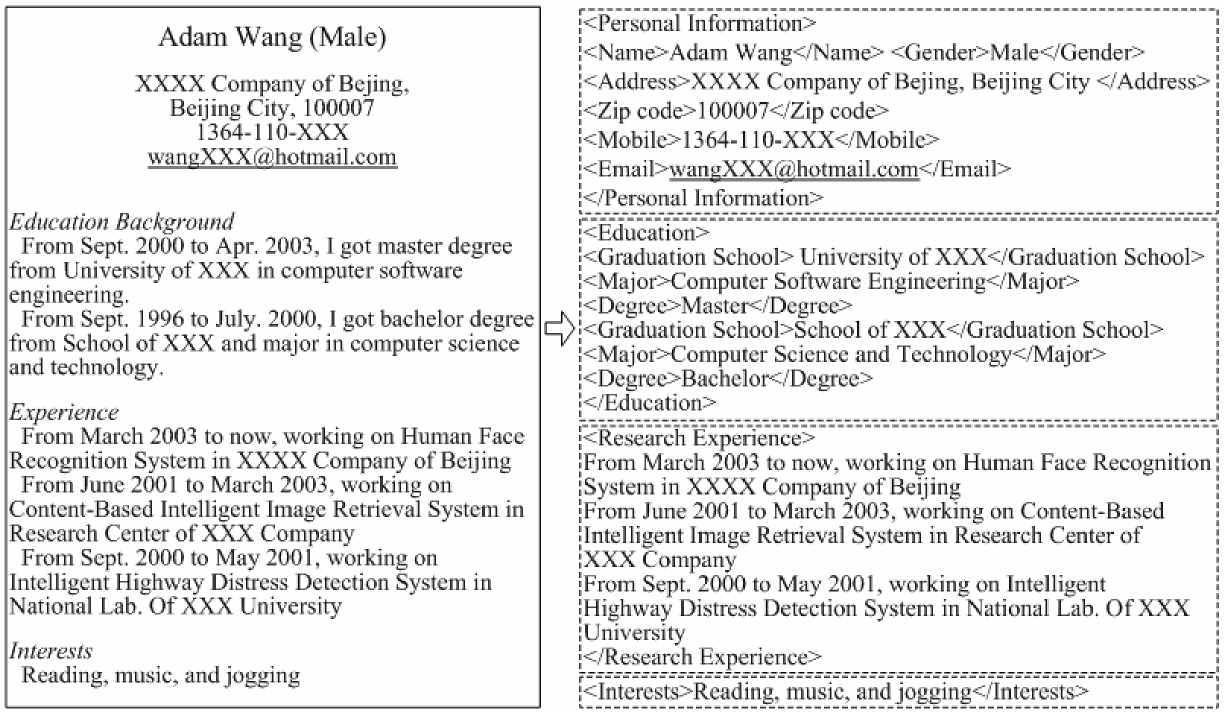
Info Hierarchy		Info Type (Label)
General Info		Personal Information(G_1); Education(G_2); Research Experience(G_3); Award(G_4); Activity(G_5); Interests(G_6); Skill(G_7)
Detailed Info	Personal Detailed Info (*Personal Information*)	Name(P_1); Gender(P_2); Birthday(P_3); Address(P_4); Zip code(P_5); Phone(P_6); Mobile(P_7); Email(P_8); Registered Residence(P_9); Marriage(P_{10}); Residence(P_{11}); Graduation School(P_{12}); Degree(P_{13}); Major(P_{14})
	Educational Detailed Info (*Education*)	Graduation School(D_1); Degree(D_2); Major(D_3); Department(D_4)

Table 1. Predefined information types.

Based on the requirements of an ongoing recruitment management system which incorporates database construction with IE technologies and resume recommendation (routing), as shown in Table 1, 7 general information fields are defined. Then, for *Personal Information*, 14 detailed information fields are designed; for *Education*, 4 detailed information fields are designed. The IE task, as exemplified in Figure 1, includes segmenting a resume into consecutive blocks labelled with general information types, and further extracting the detailed information such as *Name* and *Address* from certain blocks.

Extracting information from resumes with high precision and recall is not an easy task. In spite of

The research was carried out in Microsoft Research Asia.


Adam Wang (Male)

XXXX Company of Bejing,
Beijing City, 100007
1364-110-XXX
wangXXX@hotmail.com

Education Background
From Sept. 2000 to Apr. 2003, I got master degree from University of XXX in computer software engineering.
From Sept. 1996 to July. 2000, I got bachelor degree from School of XXX and major in computer science and technology.

Experience
From March 2003 to now, working on Human Face Recognition System in XXXX Company of Beijing
From June 2001 to March 2003, working on Content-Based Intelligent Image Retrieval System in Research Center of XXX Company
From Sept. 2000 to May 2001, working on Intelligent Highway Distress Detection System in National Lab. Of XXX University

Interests
Reading, music, and jogging

<Personal Information>
<Name>Adam Wang</Name> <Gender>Male</Gender>
<Address>XXXX Company of Bejing, Beijing City </Address>
<Zip code>100007</Zip code>
<Mobile>1364-110-XXX</Mobile>
<Email>wangXXX@hotmail.com</Email>
</Personal Information>

<Education>
<Graduation School> University of XXX</Graduation School>
<Major>Computer Software Engineering</Major>
<Degree>Master</Degree>
<Graduation School>School of XXX</Graduation School>
<Major>Computer Science and Technology</Major>
<Degree>Bachelor</Degree>
</Education>

<Research Experience>
From March 2003 to now, working on Human Face Recognition System in XXXX Company of Beijing
From June 2001 to March 2003, working on Content-Based Intelligent Image Retrieval System in Research Center of XXX Company
From Sept. 2000 to May 2001, working on Intelligent Highway Distress Detection System in National Lab. Of XXX University
</Research Experience>

<Interests>Reading, music, and jogging</Interests>

Figure 1. Example of a resume and the extracted information.

constituting a restricted domain, resumes can be written in multitude of formats (e.g. structured tables or plain texts), in different languages (e.g. Chinese and English) and in different file types (e.g. Text, PDF, Word etc.). Moreover, writing styles could be very diversified.

Among the methods in IE, Hidden Markov modelling has been widely used (Freitag and McCallum, 1999; Borkar et al., 2001). As a state-based model, HMMs are good at extracting information fields that hold a strong order of sequence. Classification is another popular method in IE. By assuming the independence of information types, it is feasible to classify segmented units as either information types to be extracted (Kushmerick et al., 2001; Peshkin and Pfeffer, 2003; Sitter and Daelemans, 2003), or information boundaries (Finn and Kushmerick, 2004). This method specializes in settling the extraction problem of independent information types.

Resume shares a document-level hierarchical contextual structure where the related information units usually occur in the same textual block, and text blocks of different information categories usually occur in a relatively fixed order. Such characteristics have been successfully used in the categorization of multi-page documents by Frasconi et al. (2001).

In this paper, given the hierarchy of resume information, a cascaded two-pass IE framework is designed. In the first pass, the general information is extracted by segmenting the entire resume into consecutive blocks and each block is annotated with a label indicating its category. In the second pass, detailed information pieces are further extracted within the boundary of certain blocks. Moreover, for different types of information, the most appropriate extraction method is selected through experiments. For the first pass, since there exists a strong sequence among blocks, a HMM model is applied to segment a resume and each block is labelled with a category of general information. We also apply HMM for the educational detailed information extraction for the same reason. In addition, classification based method is selected for the personal detailed information extraction where information items appear relatively independently.

Tested with 1,200 Chinese resumes, experimental results show that exploring the hierarchical structure of resumes with this proposed cascaded framework improves the average F-score of detailed information extraction greatly, and combining different IE models in different layer properly is effective to achieve good precision and recall.

The remaining part of this paper is structured as follows. Section 2 introduces the related work. Section 3 presents the structure of the cascaded hybrid IE model and introduces the HMM model and SVM model in detail. Experimental results and analysis are shown in Section 4. Section 5 provides a discussion of our cascaded hybrid model. Section 6 is the conclusion and future work.

2 Related Work

As far as we know, there are few published works on resume IE except some products, for which there is no way to determine the technical details. One of the published results on resume IE was shown in Ciravegna and Lavelli (2004). In this work, they applied $(LP)^2$, a toolkit of IE, to learn information extraction rules for resumes written in English. The information defined in their task includes a flat structure of *Name*, *Street*, *City*, *Province*, *Email*, *Telephone*, *Fax* and *Zip code*. This flat setting is not only different from our hierarchical structure but also different from our detailed information pieces.

Besides, there are some applications that are analogous to resume IE, such as seminar announcement IE (Freitag and McCallum, 1999), job posting IE (Sitter and Daelemans, 2003; Finn and Kushmerick, 2004) and address segmentation (Borkar et al., 2001; Kushmerick et al., 2001). Most of the approaches employed in these applications view a text as flat and extract information from all the texts directly (Freitag and McCallum, 1999; Kushmerick et al., 2001; Peshkin and Pfeffer, 2003; Finn and Kushmerick, 2004). Only a few approaches extract information hierarchically like our model. Sitter and Daelemans (2003) present a double classification approach to perform IE by extracting words from pre-extracted sentences. Borkar et al. (2001) develop a nested model, where the outer HMM captures the sequencing relationship among elements and the inner HMMs learn the finer structure within each element. But these approaches employ the same IE methods for all the information types. Compared with them, our model applies different methods in different sub-tasks to fit the special contextual structure of information in each sub-task well.

3 Cascaded Hybrid Model

Figure 2 is the structure of our cascaded hybrid model. The first pass (on the left hand side) segments a resume into consecutive blocks with a HMM model. Then based on the result, the second pass (on the right hand side) uses HMM to extract the educational detailed information and SVM to extract the personal detailed information, respectively. The block selection module is used to decide the range of detailed information extraction in the second pass.

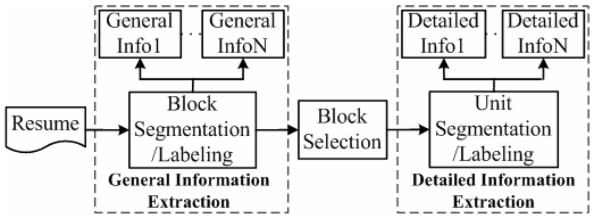

General Info1
General InfoN
Detailed Info1
Detailed InfoN
Resume
Block Segmentation /Labeling
Block Selection
Unit Segmentation /Labeling
General Information Extraction
Detailed Information Extraction

Figure 2. Structure of cascaded hybrid model.

3.1 HMM Model

3.1.1 Model Design

For general information, the IE task is viewed as labelling the segmented units with predefined class labels. Given an input resume T which is a sequence of words $w_1,w_2,\ldots,w_k$, the result of general information extraction is a sequence of blocks in which some words are grouped into a certain block $T = t_1, t_2,\ldots, t_n$, where t_i is a block. Assuming the expected label sequence of T is $L=l_1, l_2,\ldots, l_n$, with each block being assigned a label l_i, we get the sequence of block and label pairs $Q=(t_1, l_1), (t_2, l_2),\ldots,(t_n, l_n)$. In our research, we simply assume that the segmentation is based on the natural paragraph of T.

Table 1 gives the list of information types to be extracted, where general information is represented as G_1~G_7. For each kind of general information, say G_i, two labels are set: G_i-B means the beginning of G_i, G_i-M means the remainder part of G_i. In addition, label O is defined to represent a block that does not belong to any general information types. With these positional information labels, general information can be obtained. For instance, if the label sequence Q for a resume with 10 paragraphs is $Q=(t_1, G_1\text{-}B)$, $(t_2, G_1\text{-}M)$, $(t_3, G_2\text{-}B)$, $(t_4, G_2\text{-}M)$, $(t_5, G_2\text{-}M)$, (t_6, O) , (t_7, O) , $(t_8, G_3\text{-}B)$, $(t_9, G_3\text{-}M)$, $(t_{10}, G_3\text{-}M)$, three types of general information can be extracted as follows: G_1:$[t_1, t_2]$, G_2:$[t_3, t_4, t_5]$, G_3:$[t_8, t_9, t_{10}]$.

Formally, given a resume $T=t_1,t_2,\ldots,t_n$, seek a label sequence $L^*=l_1,l_2,\ldots,l_n$, such that the probability of the sequence of labels is maximal.

$$L^* = \arg\max_L P(L \mid T) \tag{1}$$

According to Bayes' equation, we have

$$L^* = \arg\max_L P(T \mid L) \times P(L) \tag{2}$$

If we assume the independent occurrence of blocks labelled as the same information types, we have

$$P(T \mid L) = \prod_{i=1}^{n} P(t_i \mid l_i) \tag{3}$$

We assume the independence of words occurring in t_i and use a unigram model, which multiplies the probabilities of these words to get the probability of t_i.

$$P(t_i \mid l_i) = \prod_{r=1}^{m} P(w_r \mid l_i), \text{where } t_i = \{w_1, w_2, \ldots w_m\} \tag{4}$$

If a tri-gram model is used to estimate $P(L)$, we have

$$P(L) = P(l_1) \times P(l_2 \mid l_1) \prod_{i=3}^{n} P(l_i \mid l_{i-1}, l_{i-2}) \tag{5}$$

To extract educational detailed information from *Education* general information, we use another HMM. It also uses two labels $D_i\text{-}B$ and $D_i\text{-}M$ to represent the beginning and remaining part of D_i, respectively. In addition, we use label O to represent that the corresponding word does not belong to any kind of educational detailed information. But this model expresses a text T as word sequence $T=w_1,w_2,\ldots,w_n$. Thus in this model, the probability $P(L)$ is calculated with Formula 5 and the probability $P(T|L)$ is calculated by

$$P(T \mid L) = \prod_{i=1}^{n} P(w_i \mid l_i) \tag{6}$$

Here we assume the independent occurrence of words labelled as the same information types.

3.1.2 Parameter Estimation

Both words and named entities are used as features in our HMMs. A Chinese resume $C= c_1', c_2', \ldots, c_k'$ is first tokenized into $C= w_1,w_2,\ldots,w_k$ with a Chinese word segmentation system LSP (Gao et al., 2003). This system outputs predefined features, including words and named entities in 8 types (Name, Date, Location, Organization, Phone, Number, Period, and Email). The named entities of the same type are normalized into single ID in feature set.

In both HMMs, fully connected structure with one state representing one information label is applied due to its convenience. To estimate the probabilities introduced in 3.1.1, maximum likelihood estimation is used, which are

$$P(l_i \mid l_{i-1}, l_{i-2}) = \frac{count(l_i, l_{i-1}, l_{i-2})}{count(l_{i-1}, l_{i-2})} \tag{7}$$

$$P(l_i \mid l_{i-1}) = \frac{count(l_i, l_{i-1})}{count(l_{i-1})} \tag{8}$$

$$P(w_r \mid l_i) = \frac{count(w_r, l_i)}{\sum_{r=1}^{m} count(w_r, l_i)}, \tag{9}$$

where state i contains m distinct words

3.1.3 Smoothing

Short of training data to estimate probability is a big problem for HMMs. Such problems may occur when estimating either $P(T|L)$ with unknown word w_i or $P(L)$ with unknown events.

Bikel et al. (1999) mapped all unknown words to one token _UNK_ and then used a held-out data to train the bi-gram models where unknown words occur. They also applied a back-off strategy to solve the data sparseness problem when estimating the context model with unknown events, which interpolates the estimation from training corpus and the estimation from the back-off model with calculated parameter λ (Bikel et al., 1999). Freitag and McCallum (1999) used shrinkage to estimate the emission probability of unknown words, which combines the estimates from data-sparse states of the complex model and the estimates in related data-rich states of the simpler models with a weighted average.

In our HMMs, we first apply Good Turing smoothing (Gale, 1995) to estimate the probability $P(w_r|l_i)$ when training data is sparse. For word w_r seen in training data, the emission probability is $P(w_r|l_i)\times(1\text{-}x)$, where $P(w_r|l_i)$ is the emission probability calculated with Formula 9 and $x=E_i/S_i$ (E_i is the number of words appearing only once in state i and S_i is the total number of words occurring in state i). For unknown word w_r, the emission probability is $x/(M\text{-}m_i)$, where M is the number of all the words appearing in training data, and m_i is the number of distinct words occurring in state i. Then, we use a back-off schema (Katz, 1987) to deal with the data sparseness problem when estimating the probability $P(L)$ (Gao et al., 2003).

3.2 SVM Model

3.2.1 Model Design

We convert personal detailed information extraction into a classification problem. Here we select SVM as the classification model because of its robustness to over-fitting and high performance (Sebastiani, 2002). In the SVM model, the IE task is also defined as labelling segmented units with predefined class labels. We still use two labels to represent personal detailed information P_i: P_i-B represents the beginning of P_i and P_i-M represents the remainder part of P_i. Besides of that, label O means that the corresponding unit does not belong to any personal detailed information boundaries and information types. For example, for part of a resume "Name:Alice (Female)", we got three units after segmentation with punctuations, i.e. "Name", "Alice", "Female". After applying SVM classification, we can get the label sequence as P_1-B,P_1-M,P_2-B. With this sequence of unit and label pairs, two types of personal detailed information can be extracted as P_1: [Name:Alice] and P_2: [Female].

Various ways can be applied to segment T. In our work, segmentation is based on the natural sentence of T. This is based on the empirical observation that detailed information is usually separated by punctuations (e.g. comma, Tab tag or Enter tag).

The extraction of personal detailed information can be formally expressed as follows: given a text $T=t_1,t_2,\ldots,t_n$, where t_i is a unit defined by the segmenting method mentioned above, seek a label sequence $L* = l_1,l_2,\ldots,l_n$, such that the probability of the sequence of labels is maximal.

$$L^* = \arg\max_{L} P(L \mid T) \qquad (10)$$

The key assumption to apply classification in IE is the independence of label assignment between units. With this assumption, Formula 10 can be described as

$$L^* = \arg\max_{L=l_1,l_2\ldots l_n} \prod_{i=1}^{n} P(l_i \mid t_i) \qquad (11)$$

Thus this probability can be maximized by maximizing each term in turn. Here, we use the SVM score of labelling t_i with l_i to replace P($l_i|t_i$).

3.2.2 Multi-class Classification

SVM is a binary classification model. But in our IE task, it needs to classify units into N classes, where N is two times of the number of personal detailed information types. There are two popular strategies to extend a binary classification task to N classes (A.Berger, 1999). The first is *One vs. All* strategy, where N classifiers are built to separate one class from others. The other is *Pairwise* strategy, where $N\times(N$-1)/2 classifiers considering all pairs of classes are built and final decision is given by their weighted voting. In our model, we apply the *One vs. All* strategy for its good efficiency in classification. We construct one classifier for each type, and classify each unit with all these classifiers. Then we select the type that has the highest score in classification. If the selected score is higher than a predefined threshold, then the unit is labelled as this type. Otherwise it is labelled as O.

3.2.3 Feature Definition

Features defined in our SVM model are described as follows:

Word: Words that occur in the unit. Each word appearing in the dictionary is a feature. We use $TF\times IDF$ as feature weight, where TF means word frequency in the text, and IDF is defined as:

$$IDF(w) = Log_2 \frac{N}{N_w} \qquad (12)$$

N: the total number of training examples;
N_w: the total number of positive examples that contain word w

Named Entity: Similar to the HMM models, 8 types of named entities identified by LSP, i.e., Name, Date, Location, Organization, Phone, Number, Period, Email, are selected as binary features. If any one type of them appears in the text, then the weight of this feature is 1, otherwise is 0.

3.3 Block Selection

Block selection is used to select the blocks generated from the first pass as the input of the second pass for detailed information extraction.

Error analysis of preliminary experiments shows that the majority of the mistakes of general information extraction resulted from labelling non-

Model	Personal Detailed Info (SVM)			Educational Detailed Info (HMM)		
	Avg.P (%)	Avg.R (%)	Avg.F (%)	Avg.P (%)	Avg.R (%)	Avg.F (%)
Flat	77.49	82.02	77.74	58.83	77.35	66.02
Cascaded	**86.83** (+9.34)	**76.89** (-5.13)	**80.44** (+2.70)	**70.78** (+11.95)	**76.80** (-0.55)	**73.40** (+7.38)

Table 2. IE results with cascaded model and flat model.

boundary blocks as boundaries in the first pass. Therefore we apply a fuzzy block selection strategy, which not only selects the blocks labelled with target general information, but also selects their neighboring two blocks, so as to enlarge the extracting range.

4 Experiments and Analysis

4.1 Data and Experimental Setting

We evaluated this cascaded hybrid model with 1,200 Chinese resumes. The data set was divided into 3 parts: training data, parameter tuning data and testing data with the proportion of 4:1:1. 6-folder cross validation was conducted in all the experiments. We selected SVMlight (Joachims, 1999) as the SVM classifier toolkit and LSP (Gao et al., 2003) for Chinese word segmentation and named entity identification. Precision (P), recall (R) and F-score (F=2PR/(P+R)) were used as the basic evaluation metrics and macro-averaging strategy was used to calculate the average results. For the special application background of our resume IE model, the "Overlap" criterion (Lavelli et al., 2004) was used to match reference instances and extracted instances. We define that if the proportion of the overlapping part of extracted instance and reference instance is over 90%, then they match each other.

A set of experiments have been designed to verify the effectiveness of exploring document-level hierarchical structure of resume and choose the best IE models (HMM vs. classification) for each sub-task.

- Cascaded model vs. flat model

Two flat models with different IE methods (SVM and HMM) are designed to extract personal detailed information and educational detailed information respectively. In these models, no hierarchical structure is used and the detailed information is extracted from the entire resume texts rather than from specific blocks. These two flat models will be compared with our proposed cascaded model.

- Model selection for different IE tasks

Both SVM and HMM are tested for all the IE tasks in first pass and in second pass.

4.2 Cascaded Model vs. Flat Model

We tested the flat model and cascaded model with detailed information extraction to verify the effectiveness of exploring document-level hierarchical structure. Results (see Table 2) show that with the cascaded model, the precision is greatly improved compared with the flat model with identical IE method, especially for educational detailed information. Although there is some loss in recall, the average F-score is still largely improved in the cascaded model.

4.3 Model Selection for Different IE Tasks

Then we tested different models for the general information and detailed information to choose the most appropriate IE model for each sub-task.

Model	Avg.P (%)	Avg.R (%)
SVM	80.95	72.87
HMM	75.95	75.89

Table 3. General information extraction with different models.

Model	Personal Detailed Info		Educational Detailed Info	
	Avg.P (%)	Avg.R (%)	Avg.P (%)	Avg.R (%)
SVM	86.83	76.89	67.36	66.21
HMM	79.64	60.16	70.78	76.80

Table 4. Detailed information extraction with different models.

Results (see Table 3) show that compared with SVM, HMM achieves better recall. In our cascaded framework, the extraction range of detailed information is influenced by the result of general information extraction. Thus better recall of general information leads to better recall of detailed information subsequently. For this reason, we choose HMM in the first pass of our cascaded hybrid model.

Then in the second pass, different IE models are tested in order to select the most appropriate one for different sub-tasks. Results (see Table 4) show that HMM performs much better in both precision and recall than SVM for educational detailed information extraction. We think that this is reasonable because HMM takes into account the sequence constraints among educational detailed information types. Therefore HMM model is selected to extract educational detailed information in our cascaded hybrid model. While for the personal detailed information extraction, we find that the SVM model gets better precision and recall than HMM model. We think that this is because of the independent occurrence of personal detailed information. Therefore, we select SVM to extract personal detailed information in our cascaded model.

5 Discussion

Our cascaded framework is a "pipeline" approach and it may suffer from error propagation. For instance, the error in the first pass may be transferred to the second pass when determining the extraction range of detailed information. Therefore the precision and recall of detailed information extraction in the second pass may be decreased subsequently. But we are not sure whether N-Best approach (Zhai et al., 2004) would be helpful. Because our cascaded hybrid model applies different IE methods for different sub-tasks, it is difficult to incorporate the N-best strategy by either simply combining the scores of the first pass and the second pass, or using the scores of the second pass to do re-ranking to select the best results. Instead of using N-best, we apply a fuzzy block selection strategy to enlarge the search scope. Experimental results of personal detailed information extraction show that compared with the exact block selection strategy, this fuzzy strategy improves the average recall of personal detailed information from 68.48% to 71.34% and reduce the average precision from 83.27% to 81.71%. Therefore the average F-score is improved by the fuzzy strategy from 75.15% to 76.17%.

Features are crucial to our SVM model. For some fields (such as *Name*, *Address* and *Graduation School*), only using words as features may result in low accuracy in IE. The named entity (NE) features used in our model enhance the accuracy of detailed information extraction. As exemplified by the results (see Table 5) on personal detailed information extraction, after adding named entity features, the F-score are improved greatly.

Field	Word +NE (%)	Word (%)
Name	90.22	3.11
Birthday	87.31	84.82
Address	67.76	49.16
Phone	81.57	75.31
Mobile	70.64	58.01
Email	88.76	85.96
Registered Residence	75.97	72.73
Residence	51.61	42.86
Graduation School	40.96	15.38
Degree	73.20	63.16
Major	63.09	43.24

Table 5. Personal detailed information extraction with different features (Avg.F).

In our cascaded hybrid model, we apply HMM and SVM in different pass separately to explore the contextual structure of information types. It guarantees the simplicity of our hybrid model. However, there are other ways to combine state-based and discriminative ideas. For example, Peng and McCallum (2004) applied Conditional Random Fields to extract information, which draws together the advantages of both HMM and SVM. This approach could be considered in our future experiments.

Some personal detailed information types do not achieve good average F-score in our model, such as *Zip code* (74.50%) and *Mobile* (73.90%). Error analysis shows that it is because these fields do not contain distinguishing words and named entities. For example, it is difficult to extract *Mobile* from the text "Phone: 010-62617711 (13859750123)". But these fields can be easily distinguished with their internal characteristics. For example, *Mobile* often consists of certain length of digital figures. To identify these fields, the Finite-State Automaton (FSA) that employs hand-crafted grammars is very effective (Hsu and Chang, 1999). Alternatively, rules learned from annotated data are also very promising in handling this case (Ciravegna and Lavelli, 2004).

We assume the independence of words occurring in unit t_i to calculate the probability

$P(t_i|l_i)$ in HMM model. While in Bikel et al. (1999), a bi-gram model is applied where each word is conditioned on its immediate predecessor when generating words inside the current name-class. We will compare this method with our current method in the future.

6 Conclusions and Future Work

We have shown that a cascaded hybrid model yields good results for the task of information extraction from resumes. We tested different models for the first pass and the second pass, and for different IE tasks. Our experimental results show that the HMM model is effective in handling the general information extraction and educational detailed information extraction, where there exists strong sequence of information pieces. And the SVM model is effective for the personal detailed information extraction.

We hope to continue this work in the future by investigating the use of other well researched IE methods. As our future works, we will apply FSA or learned rules to improve the precision and recall of some personal detailed information (such as *Zip code* and *Mobile*). Other smoothing methods such as (Bikel et al. 1999) will be tested in order to better overcome the data sparseness problem.

7 Acknowledgements

The authors wish to thank Dr. JianFeng Gao, Dr. Mu Li, Dr. Yajuan Lv for their help with the LSP tool, and Dr. Hang Li, Yunbo Cao for their valuable discussions on classification approaches. We are indebted to Dr. John Chen for his assistance to polish the English. We want also thank Long Jiang for his assistance to annotate the training and testing data. We also thank the three anonymous reviewers for their valuable comments.

References

A.Berger. Error-correcting output coding for text classification. 1999. In *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*.

D.M.Bikel, R.Schwartz, R.M.Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1):211-231.

V.Borkar, K.Deshmukh and S.Sarawagi. 2001. Automatic segmentation of text into structured records. In *Proceedings of ACM SIGMOD Conference*. pp.175-186.

F.Ciravegna, A.Lavelli. 2004. LearningPinocchio: adaptive information extraction for real world applications. *Journal of Natural Language Engineering*, 10(2):145-165.

A.Finn and N.Kushmerick. 2004. Multi-level boundary classification for information extraction. In *Proceedings of ECML04*.

P.Frasconi, G.Soda and A.Vullo. 2001. Text categorization for multi-page documents: a hybrid Naïve Bayes HMM approach. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*. pp.11-20.

D.Freitag and A.McCallum. 1999. Information extraction with HMMs and shrinkage. In *AAAI99 Workshop on Machine Learning for Information Extraction*. pp.31-36.

W.Gale. 1995. Good-Turing smoothing without tears. *Journal of Quantitative Linguistics*, 2:217-237.

J.F.Gao, M.Li and C.N.Huang. 2003. Improved source-channel models for Chinese word segmentation. In *Proceedings of ACL03*. pp.272-279.

C.N.Hsu and C.C.Chang. 1999. Finite-state transducers for semi-structured text mining. In *Proceedings of IJCAI99 Workshop on Text Mining: Foundations, Techniques and Applications.* pp.38-49.

T.Joachims. 1999. Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

S.M.Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE ASSP*, 35(3):400-401.

N.Kushmerick, E.Johnston and S.McGuinness. 2001. Information extraction by text classification. In *IJCAI01 Workshop on Adaptive Text Extraction and Mining*.

A.Lavelli, M.E.Califf, F.Ciravegna, D.Freitag, C.Giuliano, N.Kushmerick and L.Romano. 2004. A critical survey of the methodology for IE evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*.

F.Peng and A.McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Proceedings of HLT/NAACL-2004*. pp.329-336.

L.Peshkin and A.Pfeffer. 2003. Bayesian information extraction network. In *Proceedings of IJCAI03*. pp.421-426.

F.Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1-47.

A.D.Sitter and W.Daelemans. 2003. Information extraction via double classification. In *Proceedings of ATEM03*.

L.Zhai, P.Fung, R.Schwartz, M.Carpuat and D.Wu. 2004. Using N-best lists for named entity recognition from Chinese speech. In *Proceedings of HLT/NAACL-2004*.

Discriminative Syntactic Language Modeling for Speech Recognition

Michael Collins
MIT CSAIL
mcollins@csail.mit.edu

Brian Roark
OGI/OHSU
roark@cslu.ogi.edu

Murat Saraclar
Bogazici University
murat.saraclar@boun.edu.tr

Abstract

We describe a method for discriminative training of a language model that makes use of syntactic features. We follow a *reranking* approach, where a baseline recogniser is used to produce 1000-best output for each acoustic input, and a second "reranking" model is then used to choose an utterance from these 1000-best lists. The reranking model makes use of syntactic features together with a parameter estimation method that is based on the perceptron algorithm. We describe experiments on the Switchboard speech recognition task. The syntactic features provide an additional 0.3% reduction in test–set error rate beyond the model of (Roark et al., 2004a; Roark et al., 2004b) (significant at $p < 0.001$), which makes use of a discriminatively trained n-gram model, giving a total reduction of 1.2% over the baseline Switchboard system.

1 Introduction

The predominant approach within language modeling for speech recognition has been to use an n-gram language model, within the "source-channel" or "noisy-channel" paradigm. The language model assigns a probability $P_l(\mathbf{w})$ to each string $\mathbf{w}$ in the language; the acoustic model assigns a conditional probability $P_a(\mathbf{a}|\mathbf{w})$ to each pair $(\mathbf{a}, \mathbf{w})$ where $\mathbf{a}$ is a sequence of acoustic vectors, and $\mathbf{w}$ is a string. For a given acoustic input $\mathbf{a}$, the highest scoring string under the model is

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left(\beta \log P_l(\mathbf{w}) + \log P_a(\mathbf{a}|\mathbf{w})\right) \quad (1)$$

where $\beta > 0$ is some value that reflects the relative importance of the language model; β is typically chosen by optimization on held-out data. In an n-gram language model, a Markov assumption is made, namely that each word depends only on the previous $(n-1)$ words. The parameters of the language model are usually estimated from a large quantity of text data. See (Chen and Goodman, 1998) for an overview of estimation techniques for n-gram models.

This paper describes a method for incorporating syntactic features into the language model, using discriminative parameter estimation techniques. We build on the work in Roark et al. (2004a; 2004b), which was summarized and extended in Roark et al. (2005). These papers used discriminative methods for n-gram language models. Our approach reranks the 1000-best output from the Switchboard recognizer of Ljolje et al. (2003).[1] Each candidate string $\mathbf{w}$ is parsed using the statistical parser of Collins (1999) to give a parse tree $\mathcal{T}(\mathbf{w})$. Information from the parse tree is incorporated in the model using a feature-vector approach: we define $\Phi(\mathbf{a}, \mathbf{w})$ to be a d-dimensional feature vector which in principle could track arbitrary features of the string $\mathbf{w}$ together with the acoustic input $\mathbf{a}$. In this paper we restrict $\Phi(\mathbf{a}, \mathbf{w})$ to only consider the string $\mathbf{w}$ and/or the parse tree $\mathcal{T}(\mathbf{w})$ for $\mathbf{w}$. For example, $\Phi(\mathbf{a}, \mathbf{w})$ might track counts of context-free rule productions in $\mathcal{T}(\mathbf{w})$, or bigram lexical dependencies within $\mathcal{T}(\mathbf{w})$. The optimal string under our new model is defined as

$$\begin{aligned}\mathbf{w}^* \quad = \quad & \arg\max_{\mathbf{w}} \left(\beta \log P_l(\mathbf{w}) + \langle \bar{\alpha}, \Phi(\mathbf{a}, \mathbf{w}) \rangle + \right. \\ & \left. \log P_a(\mathbf{a}|\mathbf{w})\right) \qquad (2)\end{aligned}$$

where the $\arg\max$ is taken over all strings in the 1000-best list, and where $\bar{\alpha} \in \mathbb{R}^d$ is a parameter vector specifying the "weight" for each feature in Φ (note that we define $\langle x, y \rangle$ to be the inner, or dot

[1]Note that (Roark et al., 2004a; Roark et al., 2004b) give results for an n-gram approach on this data which makes use of both lattices and 1000-best lists. The results on 1000-best lists were very close to results on lattices for this domain, suggesting that the 1000-best approximation is a reasonable one.

product, between vectors x and y). For this paper, we train the parameter vector $\bar{\alpha}$ using the perceptron algorithm (Collins, 2004; Collins, 2002). The perceptron algorithm is a very fast training method, in practice requiring only a few passes over the training set, allowing for a detailed comparison of a wide variety of feature sets.

A number of researchers have described work that incorporates syntactic language models into a speech recognizer. These methods have almost exclusively worked within the noisy channel paradigm, where the syntactic language model has the task of modeling a distribution over strings in the language, in a very similar way to traditional n-gram language models. The Structured Language Model (Chelba and Jelinek, 1998; Chelba and Jelinek, 2000; Chelba, 2000; Xu et al., 2002; Xu et al., 2003) makes use of an incremental shift-reduce parser to enable the probability of words to be conditioned on k previous c-commanding lexical heads, rather than simply on the previous k words. Incremental top-down and left-corner parsing (Roark, 2001a; Roark, 2001b) and head-driven parsing (Charniak, 2001) approaches have directly used generative PCFG models as language models. In the work of Wen Wang and Mary Harper (Wang and Harper, 2002; Wang, 2003; Wang et al., 2004), a constraint dependency grammar and a finite-state tagging model derived from that grammar were used to exploit syntactic dependencies.

Our approach differs from previous work in a couple of important respects. First, through the feature-vector representations $\Phi(\mathbf{a}, \mathbf{w})$ we can essentially incorporate arbitrary sources of information from the string or parse tree into the model. We would argue that our method allows considerably more flexibility in terms of the choice of features in the model; in previous work features were incorporated in the model through modification of the underlying generative parsing or tagging model, and modifying a generative model is a rather indirect way of changing the features used by a model. In this respect, our approach is similar to that advocated in Rosenfeld et al. (2001), which used Maximum Entropy modeling to allow for the use of shallow syntactic features for language modeling.

A second contrast between our work and previous work, including that of Rosenfeld et al. (2001), is in the use of discriminative parameter estimation techniques. The criterion we use to optimize the parameter vector $\bar{\alpha}$ is closely related to the end goal in speech recognition, i.e., word error rate. Previous work (Roark et al., 2004a; Roark et al., 2004b) has shown that discriminative methods within an n-gram approach can lead to significant reductions in WER, in spite of the features being of the same type as the original language model. In this paper we extend this approach, by including syntactic features that were not in the baseline speech recognizer.

This paper describe experiments using a variety of syntactic features within this approach. We tested the model on the Switchboard (SWB) domain, using the recognizer of Ljolje et al. (2003). The discriminative approach for n-gram modeling gave a 0.9% reduction in WER on this domain; the syntactic features we describe give a further 0.3% reduction.

In the remainder of this paper, section 2 describes previous work, including the parameter estimation methods we use, and section 3 describes the feature-vector representations of parse trees that we used in our experiments. Section 4 describes experiments using the approach.

2 Background

2.1 Previous Work

Techniques for exploiting stochastic context-free grammars for language modeling have been explored for more than a decade. Early approaches included algorithms for efficiently calculating string prefix probabilities (Jelinek and Lafferty, 1991; Stolcke, 1995) and approaches to exploit such algorithms to produce n-gram models (Stolcke and Segal, 1994; Jurafsky et al., 1995). The work of Chelba and Jelinek (Chelba and Jelinek, 1998; Chelba and Jelinek, 2000; Chelba, 2000) involved the use of a shift-reduce parser trained on Penn treebank style annotations, that maintains a weighted set of parses as it traverses the string from left-to-right. Each word is predicted by each candidate parse in this set at the point when the word is shifted, and the conditional probability of the word given the previous words is taken as the weighted sum of the conditional probabilities provided by each parse. In this approach, the probability of a word is conditioned by the top two lexical heads on the stack of the particular parse. Enhancements in the feature set and improved parameter estimation techniques have extended this approach in recent years (Xu et al., 2002; Xu et al., 2003).

Roark (2001a; 2001b) pursued a different derivation strategy from Chelba and Jelinek, and used the parse probabilities directly to calculate the string probabilities. This work made use of a left-to-right, top-down, beam-search parser, which exploits rich lexico-syntactic features from the left context of each derivation to condition derivation move probabilities, leading to a very peaked distribution. Rather than normalizing a prediction of the next word over the beam of candidates, as in Chelba and Jelinek, in this approach the string probability is derived by simply summing the probabilities of all derivations for that string in the beam.

Other work on syntactic language modeling includes that of Charniak (2001), which made use of a non-incremental, head-driven statistical parser to produce string probabilities. In the work of Wen Wang and Mary Harper (Wang and Harper, 2002; Wang, 2003; Wang et al., 2004), a constraint dependency grammar and a finite-state tagging model derived from that grammar, were used to exploit syntactic dependencies. The processing advantages of the finite-state encoding of the model has allowed for the use of probabilities calculated off-line from this model to be used in the first pass of decoding, which has provided additional benefits. Finally, Och et al. (2004) use a reranking approach with syntactic information within a machine translation system.

Rosenfeld et al. (2001) investigated the use of syntactic features in a Maximum Entropy approach. In their paper, they used a shallow parser to annotate base constituents, and derived features from sequences of base constituents. The features were indicator features that were either (1) exact matches between a set or sequence of base constituents with those annotated on the hypothesis transcription; or (2) tri-tag features from the constituent sequence. The generative model that resulted from their feature set resulted in only a very small improvement in either perplexity or word-error-rate.

2.2 Global Linear Models

We follow the framework of Collins (2002; 2004), recently applied to language modeling in Roark et al. (2004a; 2004b). The model we propose consists of the following components:

- $\mathbf{GEN}(\mathbf{a})$ is a set of candidate strings for an acoustic input $\mathbf{a}$. In our case, $\mathbf{GEN}(\mathbf{a})$ is a set of 1000-best strings from a first-pass recognizer.
- $\mathcal{T}(\mathbf{w})$ is the parse tree for string $\mathbf{w}$.
- $\Phi(\mathbf{a}, \mathbf{w}) \in \mathbb{R}^d$ is a feature-vector representation of an acoustic input $\mathbf{a}$ together with a string $\mathbf{w}$.
- $\bar{\alpha} \in \mathbb{R}^d$ is a parameter vector.
- The output of the recognizer for an input $\mathbf{a}$ is defined as

$$F(\mathbf{a}) = \underset{\mathbf{w} \in \mathbf{GEN}(\mathbf{a})}{\operatorname{argmax}} \langle \Phi(\mathbf{a}, \mathbf{w}), \bar{\alpha} \rangle \tag{3}$$

In principle, the feature vector $\Phi(\mathbf{a}, \mathbf{w})$ could take into account any features of the acoustic input $\mathbf{a}$ together with the utterance $\mathbf{w}$. In this paper we make a couple of restrictions. First, we define the first feature to be

$$\Phi_1(\mathbf{a}, \mathbf{w}) = \beta \log P_l(\mathbf{w}) + \log P_a(\mathbf{a}|\mathbf{w})$$

where $P_l(\mathbf{w})$ and $P_a(\mathbf{a}|\mathbf{w})$ are language and acoustic model scores from the baseline speech recognizer. In our experiments we kept β fixed at the value used in the baseline recogniser. It can then be seen that our model is equivalent to the model in Eq. 2. Second, we restrict the remaining features $\Phi_2(\mathbf{a}, \mathbf{w}) \ldots \Phi_d(\mathbf{a}, \mathbf{w})$ to be sensitive to the string $\mathbf{w}$ alone.[2] In this sense, the scope of this paper is limited to the language modeling problem. As one example, the language modeling features might take into account n-grams, for example through definitions such as

$$\Phi_2(\mathbf{a}, \mathbf{w}) = \text{Count of } \textit{the the} \text{ in } \mathbf{w}$$

Previous work (Roark et al., 2004a; Roark et al., 2004b) considered features of this type. In this paper, we introduce syntactic features, which may be sensitive to the parse tree for $\mathbf{w}$, for example

$$\Phi_3(\mathbf{a}, \mathbf{w}) = \text{Count of } \mathtt{S} \rightarrow \mathtt{NP\ VP} \text{ in } \mathcal{T}(\mathbf{w})$$

where $\mathtt{S} \rightarrow \mathtt{NP\ VP}$ is a context-free rule production. Section 3 describes the full set of features used in the empirical results presented in this paper.

[2] Future work may consider features of the acoustic sequence $\mathbf{a}$ together with the string $\mathbf{w}$, allowing the approach to be applied to acoustic modeling.

2.2.1 Parameter Estimation

We now describe how the parameter vector $\bar{\alpha}$ is estimated from a set of training utterances. The training set consists of examples $(\mathbf{a}_i, \mathbf{w}_i)$ for $i = 1 \dots m$, where $\mathbf{a}_i$ is the i'th acoustic input, and $\mathbf{w}_i$ is the transcription of this input. We briefly review the two training algorithms described in Roark et al. (2004b), the perceptron algorithm and global conditional log-linear models (GCLMs).

Figure 1 shows the perceptron algorithm. It is an online algorithm, which makes several passes over the training set, updating the parameter vector after each training example. For a full description of the algorithm, see Collins (2004; 2002).

A second parameter estimation method, which was used in (Roark et al., 2004b), is to optimize the log-likelihood under a log-linear model. Similar approaches have been described in Johnson et al. (1999) and Lafferty et al. (2001). The objective function used in optimizing the parameters is

$$L(\bar{\alpha}) = \sum_i \log P(\mathbf{s}_i|\mathbf{a}_i, \bar{\alpha}) - C \sum_j \alpha_j^2 \quad (4)$$

where $P(\mathbf{s}_i|\mathbf{a}_i, \bar{\alpha}) = \frac{e^{\langle \Phi(\mathbf{a}_i, \mathbf{s}_i), \bar{\alpha} \rangle}}{\sum_{\mathbf{w} \in \mathbf{GEN}(\mathbf{a}_i)} e^{\langle \Phi(\mathbf{a}_i, \mathbf{w}), \bar{\alpha} \rangle}}$.

Here, each $\mathbf{s}_i$ is the member of $\mathbf{GEN}(\mathbf{a}_i)$ which has lowest WER with respect to the target transcription $\mathbf{w}_i$. The first term in $L(\bar{\alpha})$ is the log-likelihood of the training data under a conditional log-linear model. The second term is a regularization term which penalizes large parameter values. C is a constant that dictates the relative weighting given to the two terms. The optimal parameters are defined as

$$\bar{\alpha}^* = \arg\max_{\bar{\alpha}} L(\bar{\alpha})$$

We refer to these models as global conditional log-linear models (GCLMs).

Each of these algorithms has advantages. A number of results—e.g., in Sha and Pereira (2003) and Roark et al. (2004b)—suggest that the GCLM approach leads to slightly higher accuracy than the perceptron training method. However the perceptron converges very quickly, often in just a few passes over the training set—in comparison GCLM's can take tens or hundreds of gradient calculations before convergence. In addition, the perceptron can be used as an effective feature selection technique, in that at each training example it only increments features seen on $\mathbf{s}_i$ or $\mathbf{y}_i$, effectively ignoring all other features seen on members of $\mathbf{GEN}(\mathbf{a}_i)$. For example, in the experiments in Roark et al. (2004a), the perceptron converged in around 3 passes over the training set, while picking non-zero values for around 1.4 million n-gram features out of a possible 41 million n-gram features seen in the training set.

Input: A parameter specifying the number of iterations over the training set, T. A value for the first parameter, α. A feature-vector representation $\Phi(\mathbf{a}, \mathbf{w}) \in \mathbb{R}^d$. Training examples $(\mathbf{a}_i, \mathbf{w}_i)$ for $i = 1 \dots m$. An n-best list $\mathbf{GEN}(\mathbf{a}_i)$ for each training utterance. We take $\mathbf{s}_i$ to be the member of $\mathbf{GEN}(\mathbf{a}_i)$ which has the lowest WER when compared to $\mathbf{w}_i$.

Initialization: Set $\alpha_1 = \alpha$, and $\alpha_j = 0$ for $j = 2 \dots d$.

Algorithm: For $t = 1 \dots T, i = 1 \dots m$

•Calculate $\mathbf{y}_i = \arg\max_{\mathbf{w} \in \mathbf{GEN}(\mathbf{a}_i)} \langle \Phi(\mathbf{a}_i, \mathbf{w}), \bar{\alpha} \rangle$

• For $j = 2 \dots m$, set $\bar{\alpha}_j = \bar{\alpha}_j + \Phi_j(\mathbf{a}_i, \mathbf{s}_i) - \Phi_j(\mathbf{a}_i, \mathbf{y}_i)$

Output: Either the final parameters $\bar{\alpha}$, or the averaged parameters $\bar{\alpha}_{avg}$ defined as $\bar{\alpha}_{avg} = \sum_{t,i} \bar{\alpha}^{t,i}/mT$ where $\bar{\alpha}^{t,i}$ is the parameter vector after training on the i'th training example on the t'th pass through the training data.

Figure 1: The perceptron training algorithm. Following Roark et al. (2004a), the parameter α_1 is set to be some constant α that is typically chosen through optimization over the development set. Recall that α_1 dictates the weight given to the baseline recognizer score.

For the present paper, to get a sense of the relative effectiveness of various kinds of syntactic features that can be derived from the output of a parser, we are reporting results using just the perceptron algorithm. This has allowed us to explore more of the potential feature space than we would have been able to do using the more costly GCLM estimation techniques. In future we plan to apply GLCM parameter estimation methods to the task.

3 Parse Tree Features

We tagged each candidate transcription with (1) part-of-speech tags, using the tagger documented in Collins (2002); and (2) a full parse tree, using the parser documented in Collins (1999). The models for both of these were trained on the Switchboard

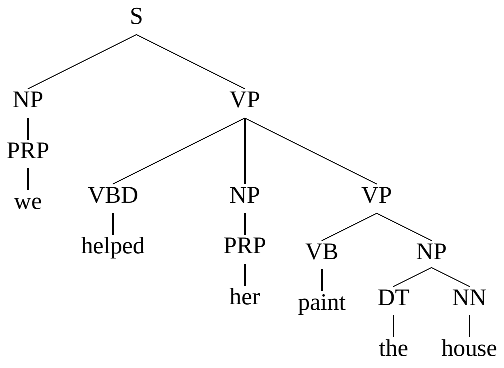


Figure 2: An example parse tree

(a)
we/PRP helped/VBD her/PRP paint/VB the/DT house/NN

(b)
we/NP^b helped/VP^b her/NP^b paint/VP^b the/NP^b house/NP^c

(c)
we/PRP-NP^b helped/VBD-VP^b her/PRP-NP^b paint/VB-VP^b the/DT-NP^b house/NN-NP^c

(d)
we/NP helped/VP her/NP paint/VP house/NP

Figure 3: Sequences derived from a parse tree: (a) POS-tag sequence; (b) Shallow parse tag sequence—the superscripts b and c refer to the beginning and continuation of a phrase respectively; (c) Shallow parse tag plus POS tag sequence; and (d) Shallow category with lexical head sequence

treebank, and applied to candidate transcriptions in both the training and test sets. Each transcription received one POS-tag annotation and one parse tree annotation, from which features were extracted.

Figure 2 shows a Penn Treebank style parse tree that is of the sort produced by the parser. Given such a structure, there is a tremendous amount of flexibility in selecting features. The first approach that we follow is to map each parse tree to sequences encoding part-of-speech (POS) decisions, and "shallow" parsing decisions. Similar representations have been used by (Rosenfeld et al., 2001; Wang and Harper, 2002). Figure 3 shows the sequential representations that we used. The first simply makes use of the POS tags for each word. The latter representations make use of sequences of non-terminals associated with lexical items. In 3(b), each word in the string is associated with the beginning or continuation of a shallow phrase or "chunk" in the tree. We include any non-terminals above the level of POS tags as potential chunks: a new "chunk" (VP, NP, PP etc.) begins whenever we see the initial word of the phrase dominated by the non-terminal. In 3(c), we show how POS tags can be added to these sequences. The final type of sequence mapping, shown in 3(d), makes a similar use of chunks, but preserves only the headword seen with each chunk.[3]

From these sequences of categories, various features can be extracted, to go along with the n-gram features used in the baseline. These include n-tag features, e.g. $t_{i-2}t_{i-1}t_i$ (where t_i represents the tag in position i); and composite tag/word features, e.g. t_iw_i (where w_i represents the word in position i) or, more complicated configurations, such as $t_{i-2}t_{i-1}w_{i-1}t_iw_i$. These features can be extracted from whatever sort of tag/word sequence we provide for feature extraction, e.g. POS-tag sequences or shallow parse tag sequences.

One variant that we performed in feature extraction had to do with how speech repairs (identified as EDITED constituents in the Switchboard style parse trees) and filled pauses or interjections (labeled with the INTJ label) were dealt with. In the simplest version, these are simply treated like other constituents in the parse tree. However, these can disrupt what may be termed the *intended* sequence of syntactic categories in the utterance, so we also tried skipping these constituents when mapping from the parse tree to shallow parse sequences.

The second set of features we employed made use of the full parse tree when extracting features. For this paper, we examined several features templates of this type. First, we considered context-free rule instances, extracted from each local node in the tree. Second, we considered features based on lexical heads within the tree. Let us first distinguish between POS-tags and non-POS non-terminal categories by calling these latter constituents NTs. For each constituent NT in the tree, there is an associated lexical head ($\mathrm{H_{NT}}$) and the POS-tag of that lexical head ($\mathrm{HP_{NT}}$). Two simple features are NT/$\mathrm{H_{NT}}$ and NT/$\mathrm{HP_{NT}}$ for every NT constituent in the tree.

[3]It should be noted that for a very small percentage of hypotheses, the parser failed to return a full parse tree. At the end of every shallow tag or category sequence, a special end of sequence tag/word pair "`</parse> </parse>`" was emitted. In contrast, when a parse failed, the sequence consisted of solely "`<noparse> <noparse>`".

Feature	Examples from figure 2
$(P,HC_P,C_i,\{+,-\}\{1,2\},H_P,H_{C_i})$	(VP,VB,NP,1,paint,house) (S,VP,NP,-1,helped,we)
$(P,HC_P,C_i,\{+,-\}\{1,2\},H_P,HP_{C_i})$	(VP,VB,NP,1,paint,NN) (S,VP,NP,-1,helped,PRP)
$(P,HC_P,C_i,\{+,-\}\{1,2\},HP_P,H_{C_i})$	(VP,VB,NP,1,VB,house) (S,VP,NP,-1,VBD,we)
$(P,HC_P,C_i,\{+,-\}\{1,2\},HP_P,HP_{C_i})$	(VP,VB,NP,1,VB,NN) (S,VP,NP,-1,VBD,PRP)

Table 1: Examples of head-to-head features. The examples are derived from the tree in figure 2.

Using the heads as identified in the parser, example features from the tree in figure 2 would be S/VBD, S/helped, NP/NN, and NP/house.

Beyond these constituent/head features, we can look at the head-to-head dependencies of the sort used by the parser. Consider each local tree, consisting of a parent node (P), a head child (HC_P), and k non-head children ($C_1 \ldots C_k$). For each non-head child C_i, it is either to the left or right of HC_P, and is either adjacent or non-adjacent to HC_P. We denote these positional features as an integer, positive if to the right, negative if to the left, 1 if adjacent, and 2 if non-adjacent. Table 1 shows four head-to-head features that can be extracted for each non-head child C_i. These features include dependencies between pairs of lexical items, between a single lexical item and the part-of-speech of another item, and between pairs of part-of-speech tags in the parse.

4 Experiments

The experimental set-up we use is very similar to that of Roark et al. (2004a; 2004b), and the extensions to that work in Roark et al. (2005). We make use of the Rich Transcription 2002 evaluation test set (rt02) as our development set, and use the Rich Transcription 2003 Spring evaluation CTS test set (rt03) as test set. The rt02 set consists of 6081 sentences (63804 words) and has three subsets: Switchboard 1, Switchboard 2, Switchboard Cellular. The rt03 set consists of 9050 sentences (76083 words) and has two subsets: Switchboard and Fisher.

The training set consists of 297580 transcribed utterances (3297579 words)[4]. For each utterance, a weighted word-lattice was produced, representing alternative transcriptions, from the ASR system. The baseline ASR system that we are comparing against then performed a rescoring pass on these first pass lattices, allowing for better silence modeling, and replaces the trigram language model score with a 6-gram model. 1000-best lists were then extracted from these lattices. For each candidate in the 1000-best lists, we identified the number of edits (insertions, deletions or substitutions) for that candidate, relative to the “target” transcribed utterance. The oracle score for the 1000-best lists was 16.7%.

[4]Note that Roark et al. (2004a; 2004b; 2005) used 20854 of these utterances (249774 words) as held out data. In this work we simply use the rt02 test set as held out and development data.

To produce the word-lattices, each training utterance was processed by the baseline ASR system. In a naive approach, we would simply train the baseline system (i.e., an acoustic model and language model) on the entire training set, and then decode the training utterances with this system to produce lattices. We would then use these lattices with the perceptron algorithm. Unfortunately, this approach is likely to produce a set of training lattices that are very different from test lattices, in that they will have very low word-error rates, given that the lattice for each utterance was produced by a model that was trained on that utterance. To somewhat control for this, the training set was partitioned into 28 sets, and baseline Katz backoff trigram models were built for each set by including only transcripts from the other 27 sets. Lattices for each utterance were produced with an acoustic model that had been trained on the entire training set, but with a language model that was trained on the 27 data portions that did not include the current utterance. Since language models are generally far more prone to overtraining than standard acoustic models, this goes a long way toward making the training conditions similar to testing conditions. Similar procedures were used to train the parsing and tagging models for the training set, since the Switchboard treebank overlaps extensively with the ASR training utterances.

Table 2 presents the word-error rates on rt02 and rt03 of the baseline ASR system, 1000-best perceptron and GCLM results from Roark et al. (2005) under this condition, and our 1000-best perceptron results. Note that our n-best result, using just n-gram features, improves upon the perceptron result of (Roark et al., 2005) by 0.2 percent, putting us within 0.1 percent of their GCLM result for that

Trial	WER	
	rt02	rt03
ASR system output	37.1	36.4
Roark et al. (2005) perceptron	36.6	35.7
Roark et al. (2005) GCLM	36.3	35.4
n-gram perceptron	36.4	35.5

Table 2: Baseline word-error rates versus Roark et al. (2005)

Trial	rt02 WER
ASR system output	37.1
n-gram perceptron	36.4
n-gram + POS (1) perceptron	36.1
n-gram + POS (1,2) perceptron	36.1
n-gram + POS (1,3) perceptron	36.1

Table 3: Use of POS-tag sequence derived features

condition. (Note that the perceptron–trained n-gram features were trigrams (i.e., $n = 3$).) This is due to a larger training set being used in our experiments; we have added data that was used as held-out data in (Roark et al., 2005) to the training set that we use.

The first additional features that we experimented with were POS-tag sequence derived features. Let t_i and w_i be the POS tag and word at position i, respectively. We experimented with the following three feature definitions:

1. $(t_{i-2}t_{i-1}t_i)$, $(t_{i-1}t_i)$, (t_i), (t_iw_i)
2. $(t_{i-2}t_{i-1}w_i)$
3. $(t_{i-2}w_{i-2}t_{i-1}w_{i-1}t_iw_i)$, $(t_{i-2}t_{i-1}w_{i-1}t_iw_i)$, $(t_{i-1}w_{i-1}t_iw_i)$, $(t_{i-1}t_iw_i)$

Table 3 summarizes the results of these trials on the held out set. Using the simple features (number 1 above) yielded an improvement beyond just n-grams, but additional, more complicated features failed to yield additional improvements.

Next, we considered features derived from shallow parsing sequences. Given the results from the POS-tag sequence derived features, for any given sequence, we simply use n-tag and tag/word features (number 1 above). The first sequence type from which we extracted features was the shallow parse tag sequence (S1), as shown in figure 3(b). Next, we tried the composite shallow/POS tag sequence (S2), as in figure 3(c). Finally, we tried extracting features from the shallow constituent sequence (S3), as shown in figure 3(d). When EDITED and INTJ nodes are ignored, we refer to this condition as S3-E. For full-parse feature extraction, we tried context-free rule features (CF) and head-to-head features (H2H), of the kind shown in table 1. Table 4 shows the results of these trials on rt02.

Trial	rt02 WER
ASR system output	37.1
n-gram perceptron	36.4
n-gram + POS perceptron	36.1
n-gram + POS + S1 perceptron	36.1
n-gram + POS + S2 perceptron	36.0
n-gram + POS + S3 perceptron	36.0
n-gram + POS + S3-E perceptron	36.0
n-gram + POS + CF perceptron	36.1
n-gram + POS + H2H perceptron	36.0

Table 4: Use of shallow parse sequence and full parse derived features

Although the single digit precision in the table does not show it, the H2H trial, using features extracted from the full parses along with n-grams and POS-tag sequence features, was the best performing model on the held out data, so we selected it for application to the rt03 test data. This yielded 35.2% WER, a reduction of 0.3% absolute over what was achieved with just n-grams, which is significant at $p < 0.001$,[5] reaching a total reduction of 1.2% over the baseline recognizer.

5 Conclusion

The results presented in this paper are a first step in examining the potential utility of syntactic features for discriminative language modeling for speech recognition. We tried two possible sets of features derived from the full annotation, as well as a variety of possible feature sets derived from shallow parse and POS tag sequences, the best of which gave a small but significant improvement beyond what was provided by the n-gram features. Future work will include a further investigation of parser–derived features. In addition, we plan to explore the alternative parameter estimation methods described in (Roark et al., 2004a; Roark et al., 2004b), which were shown in this previous work to give further improvements over the perceptron.

[5]We use the Matched Pair Sentence Segment test for WER, a standard measure of signifi cance, to calculate this p-value.

References

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proc. ACL*.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 225–231.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

Ciprian Chelba. 2000. *Exploiting Syntactic Structure for Natural Language Modeling*. Ph.D. thesis, The Johns Hopkins University.

Stanley Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report, TR-10-98, Harvard University.

Michael J. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, pages 1–8.

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 Academic Publishers, Dordrecht.

Frederick Jelinek and John Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323.

Mark Johnson, Stuart Geman, Steven Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unifi cation-based" grammars. In *Proc. ACL*, pages 535–541.

Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary Tajchman, and Nelson Morgan. 1995. Using a stochastic context-free grammar as a language model for speech recognition. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 189–192.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fi elds: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, pages 282–289, Williams College, Williamstown, MA, USA.

Andrej Ljolje, Enrico Bocchieri, Michael Riley, Brian Roark, Murat Saraclar, and Izhak Shafran. 2003. The AT&T 1xRT CTS system. In *Rich Transcription Workshop*.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of HLT-NAACL 2004*.

Brian Roark, Murat Saraclar, and Michael Collins. 2004a. Corrective language modeling for large vocabulary ASR with the perceptron algorithm. In *Proc. ICASSP*, pages 749–752.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004b. Discriminative language modeling with conditional random fi elds and the perceptron algorithm. In *Proc. ACL*.

Brian Roark, Murat Saraclar, and Michael Collins. 2005. Discriminative n-gram language modeling. *Computer Speech and Language*. *submitted*.

Brian Roark. 2001a. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Brian Roark. 2001b. *Robust Probabilistic Predictive Syntactic Processing*. Ph.D. thesis, Brown University. http://arXiv.org/abs/cs/0105019.

Ronald Rosenfeld, Stanley Chen, and Xiaojin Zhu. 2001. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. In *Computer Speech and Language*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fi elds. In *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Canada.

Andreas Stolcke and Jonathan Segal. 1994. Precise n-gram probabilities from stochastic context-free grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 74–79.

Andreas Stolcke. 1995. An effi cient probabilistic context-free parsing algorithm that computes prefi x probabilities. *Computational Linguistics*, 21(2):165–202.

Wen Wang and Mary P. Harper. 2002. The superARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proc. EMNLP*, pages 238–247.

Wen Wang, Andreas Stolcke, and Mary P. Harper. 2004. The use of a linguistically motivated language model in conversational speech recognition. In *Proc. ICASSP*.

Wen Wang. 2003. *Statistical parsing and language modeling based on constraint dependency grammar*. Ph.D. thesis, Purdue University.

Peng Xu, Ciprian Chelba, and Frederick Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 191–198.

Peng Xu, Ahmad Emami, and Frederick Jelinek. 2003. Training connectionist models for the structured language model. In *Proc. EMNLP*, pages 160–167.

A Phonotactic Language Model for Spoken Language Identification

Haizhou Li *and* **Bin Ma**
Institute for Infocomm Research
Singapore 119613
{hli,mabin}@i2r.a-star.edu.sg

Abstract

We have established a phonotactic language model as the solution to spoken language identification (LID). In this framework, we define a single set of acoustic tokens to represent the acoustic activities in the world's spoken languages. A voice tokenizer converts a spoken document into a text-like document of acoustic tokens. Thus a spoken document can be represented by a count vector of acoustic tokens and token *n*-grams in the vector space. We apply *latent semantic analysis* to the vectors, in the same way that it is applied in information retrieval, in order to capture salient phonotactics present in spoken documents. The vector space modeling of spoken utterances constitutes a paradigm shift in LID technology and has proven to be very successful. It presents a 12.4% error rate reduction over one of the best reported results on the 1996 NIST Language Recognition Evaluation database.

1 Introduction

Spoken language and written language are similar in many ways. Therefore, much of the research in spoken language identification, LID, has been inspired by text-categorization methodology. Both text and voice are generated from language dependent vocabulary. For example, both can be seen as stochastic time-sequences corrupted by a channel noise. The *n*-gram language model has achieved equal amounts of success in both tasks, e.g. *n*-character slice for text categorization by language (Cavnar and Trenkle, 1994) and Phone Recognition followed by *n*-gram Language Modeling, or PRLM (Zissman, 1996) .

Orthographic forms of language, ranging from Latin alphabet to Cyrillic script to Chinese characters, are far more unique to the language than their phonetic counterparts. From the speech production point of view, thousands of spoken languages from all over the world are phonetically articulated using only a few hundred distinctive sounds or phonemes (Hieronymus, 1994). In other words, common sounds are shared considerably across different spoken languages. In addition, spoken documents[1], in the form of digitized wave files, are far less structured than written documents and need to be treated with techniques that go beyond the bounds of written language. All of this makes the identification of spoken language based on phonetic units much more challenging than the identification of written language. In fact, the challenge of LID is inter-disciplinary, involving digital signal processing, speech recognition and natural language processing.

In general, a LID system usually has three fundamental components as follows:

1) A voice tokenizer which segments incoming voice feature frames and associates the segments with acoustic or phonetic labels, called tokens;
2) A statistical language model which captures language dependent phonetic and phonotactic information from the sequences of tokens;
3) A language classifier which identifies the language based on discriminatory characteristics of acoustic score from the voice tokenizer and phonotactic score from the language model.

In this paper, we present a novel solution to the three problems, focusing on the second and third problems from a computational linguistic perspective. The paper is organized as follows: In Section 2, we summarize relevant existing approaches to the LID task. We highlight the shortcomings of existing approaches and our attempts to address the

[1] A spoken utterance is regarded as a spoken document in this paper.

issues. In Section 3 we propose the *bag-of-sounds* paradigm to turn the LID task into a typical text categorization problem. In Section 4, we study the effects of different settings in experiments on the 1996 NIST Language Recognition Evaluation (LRE) database[2]. In Section 5, we conclude our study and discuss future work.

2 Related Work

Formal evaluations conducted by the National Institute of Science and Technology (NIST) in recent years demonstrated that the most successful approach to LID used the phonotactic content of the voice signal to discriminate between a set of languages (Singer *et al.*, 2003). We briefly discuss previous work cast in the formalism mentioned above: tokenization, statistical language modeling, and language identification. A typical LID system is illustrated in Figure 1 (Zissman, 1996), where language dependent voice tokenizers (VT) and language models (LM) are deployed in the Parallel PRLM architecture, or P-PRLM.

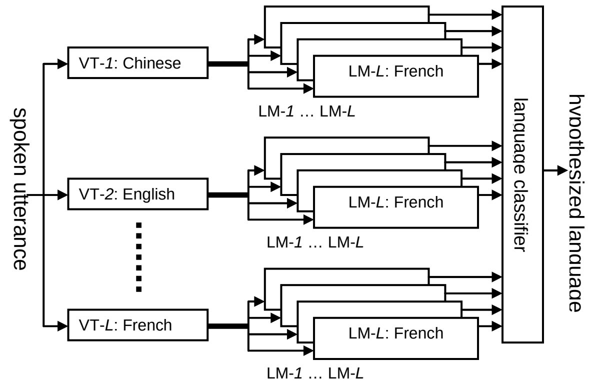


Figure 1. *L* monolingual phoneme recognition front-ends are used in parallel to tokenize the input utterance, which is analyzed by LMs to predict the spoken language

2.1 Voice Tokenization

A voice tokenizer is a speech recognizer that converts a spoken document into a sequence of tokens. As illustrated in Figure 2, a token can be of different sizes, ranging from a speech feature frame, to a phoneme, to a lexical word. A token is defined to describe a distinct acoustic/phonetic activity. In early research, low level spectral frames, which are assumed to be independent of each other, were used as a set of prototypical spectra for each language (Sugiyama, 1991). By adopting hidden Markov models, people moved beyond low-level spectral analysis towards modeling a frame sequence into a larger unit such as a phoneme and even a lexical word.

Since the lexical word is language specific, the phoneme becomes the natural choice when building a language-independent voice tokenization front-end. Previous studies show that parallel language-dependent phoneme tokenizers effectively serve as the tokenization front-ends with P-PRLM being the typical example. However, a language-independent phoneme set has not been explored yet experimentally. In this paper, we would like to explore the potential of voice tokenization using a unified phoneme set.

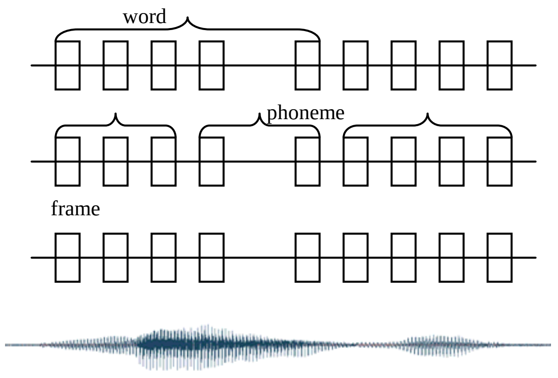


Figure 2 Tokenization at different resolutions

2.2 *n*-gram Language Model

With the sequence of tokens, we are able to estimate an *n*-gram language model (LM) from the statistics. It is generally agreed that phonotactics, i.e. the rules governing the phone/phonemes sequences admissible in a language, carry more language discriminative information than the phonemes themselves. An *n*-gram LM over the tokens describes well *n*-local phonotactics among neighboring tokens. While some systems model the phonotactics at the frame level (Torres-Carrasquillo *et al.*, 2002), others have proposed P-PRLM. The latter has become one of the most promising solutions so far (Zissman, 1996).

A variety of cues can be used by humans and machines to distinguish one language from another. These cues include phonology, prosody, morphology, and syntax in the context of an utterance.

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

However, global phonotactic cues at the level of utterance or spoken document remains unexplored in previous work. In this paper, we pay special attention to it. A spoken language always contains a set of high frequency function words, prefixes, and suffixes, which are realized as phonetic token substrings in the spoken document. Individually, those substrings may be shared across languages. However, the pattern of their co-occurrences discriminates one language from another.

Perceptual experiments have shown (Muthusamy, 1994) that with adequate training, human listeners' language identification ability increases when given longer excerpts of speech. Experiments have also shown that increased exposure to each language and longer training sessions improve listeners' language identification performance. Although it is not entirely clear how human listeners make use of the high-order phonotactic/prosodic cues present in longer spans of a spoken document, strong evidence shows that phonotactics over larger context provides valuable LID cues beyond *n*-gram, which will be further attested by our experiments in Section 4.

2.3 Language Classifier

The task of a language classifier is to make good use of the LID cues that are encoded in the model λ_l to hypothesize $\hat{l}$ from among L languages, Λ, as the one that is actually spoken in a spoken document O. The LID model λ_l in P-PRLM refers to extracted information from acoustic model and *n*-gram LM for language *l*. We have $\lambda_l = \{\lambda_l^{AM}, \lambda_l^{LM}\}$ and $\lambda_l \in \Lambda \ (l = 1, ..., L)$. A maximum-likelihood classifier can be formulated as follows:

$$\begin{aligned}\hat{l} &= \arg\max_{l \in \Lambda} P(O/\lambda_l) \\ &\approx \arg\max_{l \in \Lambda} \sum_{T \in \Gamma} P\left(O/T, \lambda_l^{AM}\right) P\left(T/\lambda_l^{LM}\right)\end{aligned} \quad (1)$$

The exact computation in Eq.(1) involves summing over all possible decoding of token sequences $T \in \Gamma$ given O. In many implementations, it is approximated by the maximum over all sequences in the sum by finding the most likely token sequence, $\hat{T}_l$, for each language *l*, using the Viterbi algorithm:

$$\hat{l} \approx \arg\max_{l \in \Lambda} [P\left(O/\hat{T}_l, \lambda_l^{AM}\right) P\left(\hat{T}_l/\lambda_l^{LM}\right)] \quad (2)$$

Intuitively, individual sounds are heavily shared among different spoken languages due to the common speech production mechanism of humans. Thus, the acoustic score has little language discriminative ability. Many experiments (Yan and Barnard, 1995; Zissman, 1996) have further attested that the *n*-gram LM score provides more language discriminative information than their acoustic counterparts. In Figure 1, the decoding of voice tokenization is governed by the acoustic model λ_l^{AM} to arrive at an acoustic score $P\left(O/\hat{T}_l, \lambda_l^{AM}\right)$ and a token sequence $\hat{T}_l$. The *n*-gram LM derives the *n*-local phonotactic score $P\left(\hat{T}_l/\lambda_l^{LM}\right)$ from the language model λ_l^{LM}.

Clearly, the *n*-gram LM suffers the major shortcoming of having not exploited the global phonotactics in the larger context of a spoken utterance. Speech recognition researchers have so far chosen to only use *n*-gram local statistics for primarily pragmatic reasons, as this *n*-gram is easier to attain. In this work, a language independent voice tokenization front-end is proposed, that uses a unified acoustic model λ^{AM} instead of multiple language dependent acoustic models λ_l^{AM}. The *n*-gram LM λ_l^{LM} is generalized to model both local and global phonotactics.

3 *Bag-of-Sounds* Paradigm

The *bag-of-sounds* concept is analogous to the *bag-of-words* paradigm originally formulated in the context of information retrieval (IR) and text categorization (TC) (Salton 1971; Berry *et al.*, 1995; Chu-Caroll and Carpenter, 1999). One focus of IR is to extract informative features for document representation. The *bag-of-words* paradigm represents a document as a vector of counts. It is believed that it is not just the words, but also the co-occurrence of words that distinguish semantic domains of text documents.

Similarly, it is generally believed in LID that, although the sounds of different spoken languages overlap considerably, the phonotactics differentiates one language from another. Therefore, one can easily draw the analogy between an acoustic token in *bag-of-sounds* and a word in *bag-of-words*. Unlike words in a text document, the phonotactic information that distinguishes spoken languages is concealed in the sound waves of spoken languages. After transcribing a spoken document into a text like document of tokens, many IR or TC techniques can then be readily applied.

It is beyond the scope of this paper to discuss what would be a good voice tokenizer. We adopt phoneme size language-independent acoustic tokens to form a unified acoustic vocabulary in our voice tokenizer. Readers are referred to (Ma *et al.*, 2005) for details of acoustic modeling.

3.1 Vector Space Modeling

In human languages, some words invariably occur more frequently than others. One of the most common ways of expressing this idea is known as Zipf's Law (Zipf, 1949). This law states that there is always a set of words which dominates most of the other words of the language in terms of their frequency of use. This is true both of written words and of spoken words. The short-term, or *local phonotactics*, is devised to describe Zipf's Law.

The local phonotactic constraints can be typically described by the token *n*-grams, or phoneme *n*-grams as in (Ng *et al*., 2000), which represents short-term statistics such as lexical constraints. Suppose that we have a token sequence, *t1 t2 t3 t4*. We derive the unigram statistics from the token sequence itself. We derive the bigram statistics from *t1(t2) t2(t3) t3(t4) t4(#)* where the token vocabulary is expanded over the token's right context. Similarly, we derive the trigram statistics from the *t1(#,t2) t2(t1,t3) t3(t2,t4) t4(t3,#)* to account for left and right contexts. The # sign is a place holder for free context. In the interest of manageability, we propose to use up to token trigram. In this way, for an acoustic system of Y tokens, we have potentially Y^2 bigram and Y^3 trigram in the vocabulary.

Meanwhile, motivated by the ideas of having both short-term and long-term phonotactic statistics, we propose to derive *global phonotactics* information to account for long-term phonotactics:

The global phonotactic constraint is the high-order statistics of *n*-grams. It represents document level long-term phonotactics such as co-occurrences of *n*-grams. By representing a spoken document as a count vector of *n*-grams, also called *bag-of-sounds* vector, it is possible to explore the relations and higher-order statistics among the diverse *n*-grams through *latent semantic analysis* (LSA).

It is often advantageous to weight the raw counts to refine the contribution of each *n*-gram to LID. We begin by normalizing the vectors representing the spoken document by making each vector of unit length. Our second weighting is based on the notion that an *n*-gram that only occurs in a few languages is more discriminative than an *n*-gram that occurs in nearly every document. We use the *inverse-document frequency* (*idf*) weighting scheme (Spark Jones, 1972), in which a word is weighted inversely to the number of documents in which it occurs, by means of $idf(w) = \log D / d(w)$, where w is a word in the vocabulary of W token *n*-grams. D is the total number of documents in the training corpus from L languages. Since each language has at least one document in the training corpus, we have $D \geq L$. $d(w)$ is the number of documents containing the word w. Letting $c_{w,d}$ be the count of word w in document d, we have the weighted count as

$$c'_{w,d} = c_{w,d} \times idf(w) / (\sum_{1 \leq w' \leq W} c^2_{w',d})^{1/2} \quad (3)$$

and a vector $c_d = \{c'_{1,d}, c'_{2,d}, ..., c'_{W,d}\}^T$ to represent document d. A corpus is then represented by a *term-document* matrix $H = \{c_1, c_2, ..., c_D\}$ of $W \times D$.

3.2 Latent Semantic Analysis

The fundamental idea in LSA is to reduce the dimension of a document vector, W to Q, where $Q << W$ and $Q << D$, by projecting the problem into the space spanned by the rows of the closest rank-Q matrix to H in the Frobenius norm (Deerwester *et al*, 1990). Through singular value decomposition (SVD) of H, we construct a modified matrix H_Q from the Q-largest singular values:

$$H_Q = U_Q S_Q V_Q^T \quad (4)$$

U_Q is a $W \times Q$ left singular matrix with rows $u_w, 1 \leq w \leq W$; S_Q is a $Q \times Q$ diagonal matrix of Q-largest singular values of $H; V_Q$ is $D \times Q$ right singular matrix with rows v_d, $1 \leq d \leq D$.

With the SVD, we project the D document vectors in H into a reduced space V_Q, referred to as Q-space in the rest of this paper. A test document c_p of unknown language ID is mapped to a pseudo-document v_p in the Q-space by matrix U_Q

$$c_p \rightarrow v_p = c_p^T U_Q S_Q^{-1} \quad (5)$$

After SVD, it is straightforward to arrive at a natural metric for the closeness between two spoken documents v_i and v_j in *Q*-space instead of their original *W*-dimensional space c_i and c_j.

$$g(c_i, c_j) \approx \cos(v_i, v_j) = \frac{v_i \cdot v_j^T}{\| v_i \| \cdot \| v_j \|} \quad (6)$$

$g(c_i, c_j)$ indicates the similarity between two vectors, which can be transformed to a distance measure $k(c_i, c_j) = \cos^{-1} g(c_i, c_j)$.

In the forced-choice classification, a test document, supposedly monolingual, is classified into one of the *L* languages. Note that the test document is unknown to the *H* matrix. We assume consistency between the test document's intrinsic phonotactic pattern and one of the *D* patterns, that is extracted from the training data and is presented in the *H* matrix, so that the SVD matrices still apply to the test document, and Eq.(5) still holds for dimension reduction.

3.3 *Bag-of-Sounds* Language Classifier

The *bag-of-sounds* phonotactic LM benefits from several properties of vector space modeling and LSA.

1) It allows for representing a spoken document as a vector of *n*-gram features, such as unigram, bigram, trigram, and the mixture of them;
2) It provides a well-defined distance metric for measurement of phonotactic distance between spoken documents;
3) It processes spoken documents in a lower dimensional *Q*-space, that makes the *bag-of-sounds* phonotactic language modeling, λ_l^{LM}, and classification computationally manageable.

Suppose we have only one prototypical vector c_l and its projection in the *Q*-space v_l to represent language *l*. Applying LSA to the *term-document* matrix $H: W \times L$, a minimum distance classifier is formulated:

$$\hat{l} = \arg\min_{l \in \Lambda} k(v_p, v_l) \quad (7)$$

In Eq.(7), v_p is the *Q*-space projection of c_p, a test document.

Apparently, it is very restrictive for each language to have just one prototypical vector, also referred to as a centroid. The pattern of language distribution is inherently multi-modal, so it is unlikely well fitted by a single vector. One solution to this problem is to span the language space with multiple vectors. Applying LSA to a *term-document* matrix $H: W \times L'$, where $L' = L \times M$ assuming each language *l* is represented by a set of *M* vectors, Φ_l, a new classifier, using *k*-nearest neighboring rule (Duda and Hart, 1973) , is formulated, named *k*-nearest classifier (KNC):

$$\hat{l} = \arg\min_{l \in \Lambda} \sum_{l' \in \phi_l} k(v_p, v_{l'}) \quad (8)$$

where ϕ_l is the set of *k*-nearest-neighbor to v_p and $\phi_l \subset \Phi_l$.

Among many ways to derive the *M* centroid vectors, here is one option. Suppose that we have a set of training documents D_l for language *l* , as subset of corpus Ω, $D_l \subset \Omega$ and $\cup_{l=1}^{L} D_l = \Omega$. To derive the *M* vectors, we choose to carry out vector quantization (VQ) to partition D_l into *M* cells $D_{l,m}$ in the *Q*-space such that $\cup_{m=1}^{M} D_{l,m} = D_l$ using similarity metric Eq.(6). All the documents in each cell $D_{l,m}$ can then be merged to form a super-document, which is further projected into a *Q*-space vector $v_{l,m}$. This results in *M* prototypical centroids $v_{l,m} \in \Phi_l$ $(m = 1, \ldots M)$. Using KNC, a test vector is compared with *M* vectors to arrive at the *k*-nearest neighbors for each language, which can be computationally expensive when *M* is large.

Alternatively, one can account for multi-modal distribution through finite mixture model. A mixture model is to represent the *M* discrete components with soft combination. To extend the KNC into a statistical framework, it is necessary to map our distance metric Eq.(6) into a probability measure. One way is for the distance measure to induce a family of exponential distributions with pertinent marginality constraints. In practice, what we need is a reasonable probability distribution, which sums to one, to act as a lookup table for the distance measure. We here choose to use the empirical multivariate distribution constructed by allocating the total probability mass in proportion to the distances observed with the training data. In short, this reduces the task to a *histogram normalization*. In this way, we map the distance $k(c_i, c_j)$ to a conditional probability distribution $p(v_i \mid v_j)$

subject to $\sum_{i=1}^{|\Omega|} p(v_i \mid v_j) = 1$. Now that we are in the probability domain, techniques such as mixture smoothing can be readily applied to model a language class with finer fitting.

Let's re-visit the task of L language forced-choice classification. Similar to KNC, suppose we have M centroids $v_{l,m} \in \Phi_l$ $(m = 1, ...M)$ in the Q-space for each language l. Each centroid represents a class. The class conditional probability can be described as a linear combination of $p(v_i \mid v_{l,m})$:

$$p(v_i \mid \lambda_l^{LM}) = \sum_{m=1}^{M} p(v_{l,m}) p(v_i \mid v_{l,m}) \qquad (9)$$

the probability $p(v_{l,m})$, functionally serves as a mixture weight of $p(v_i \mid v_{l,m})$. Together with a set of centroids $v_{l,m} \in \Phi_l$ $(m = 1, ...M)$, $p(v_i \mid v_{l,m})$ and $p(v_{l,m})$ define a mixture model λ_l^{LM}. $p(v_i \mid v_{l,m})$ is estimated by *histogram normalization* and $p(v_{l,m})$ is estimated under the maximum likelihood criteria, $p(v_{l,m}) = C_{m,l} / C_l$, where C_l is total number of documents in D_l, of which $C_{m,l}$ documents fall into the cell m.

An *Expectation-Maximization* iterative process can be devised for training of λ_l^{LM} to maximize the likelihood Eq.(9) over the entire training corpus:

$$p(\Omega \mid \Lambda) = \prod_{l=1}^{L} \prod_{d=1}^{|D_l|} p(v_d \mid \lambda_l^{LM}) \qquad (10)$$

Using the phonotactic LM score $P\left(\hat{T}_l / \lambda_l^{LM}\right)$ for classification, with $\hat{T}_l$ being represented by the *bag-of-sounds* vector v_p, Eq.(2) can be reformulated as Eq.(11), named mixture-model classifier (MMC):

$$\begin{aligned} \hat{l} &= \arg\max_{l \in \Lambda} p(v_p \mid \lambda_l^{LM}) \\ &= \arg\max_{l \in \Lambda} \sum_{m=1}^{M} p(v_{l,m}) p(v_p \mid v_{l,m}) \end{aligned} \qquad (11)$$

To establish fair comparison with P-PRLM, as shown in Figure 3, we devise our *bag-of-sounds* classifier to solely use the LM score $P\left(\hat{T}_l / \lambda_l^{LM}\right)$ for classification decision whereas the acoustic score $P\left(O / \hat{T}_l, \lambda_l^{AM}\right)$ may potentially help as reported in (Singer *et al.*, 2003).

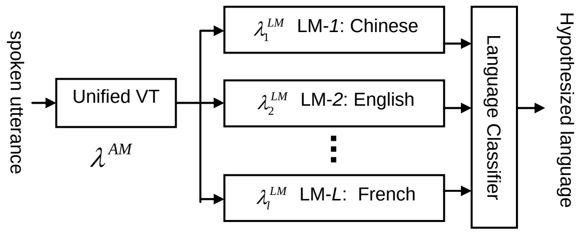

spoken utterance
Unified VT
λ^{AM}
λ_1^{LM} LM-*1*: Chinese
λ_2^{LM} LM-*2*: English
λ_l^{LM} LM-*L*: French
Language Classifier
Hypothesized language

Figure 3. A *bag-of-sounds* classifier. A unified front-end followed by L parallel *bag-of-sounds* phonotactic LMs.

4 Experiments

This section will experimentally analyze the performance of the proposed *bag-of-sounds* framework using the 1996 NIST Language Recognition Evaluation (LRE) data. The database was intended to establish a baseline of performance capability for language recognition of conversational telephone speech. The database contains recorded speech of 12 languages: Arabic, English, Farsi, French, German, Hindi, Japanese, Korean, Mandarin, Spanish, Tamil and Vietnamese. We use the training set and development set from LDC *CallFriend* corpus[3] as the training data. Each conversation is segmented into overlapping sessions of about 30 seconds each, resulting in about 12,000 sessions for each language. The evaluation set consists of 1,492 30-sec sessions, each distributed among the various languages of interest. We treat a 30-sec session as a spoken document in both training and testing. We report error rates (ER) of the 1,492 test trials.

4.1 Effect of Acoustic Vocabulary

The choice of n-gram affects the performance of LID systems. Here we would like to see how a better choice of acoustic vocabulary can help convert a spoken document into a phonotactically discriminative space. There are two parameters that determine the acoustic vocabulary: the choice of acoustic token, and the choice of n-grams. In this paper, the former concerns the size of an acoustic system Y in the unified front-end. It is studied in more details in (Ma *et al.*, 2005). We set Y to 32 in

[3] See http://www.ldc.upenn.edu/. The overlap between 1996 NIST evaluation data and *CallFriend* database has been removed from training data as suggested in the 2003 NIST LRE website http://www.nist.gov/speech/tests/index.htm

this experiment; the latter decides what features to be included in the vector space. The vector space modeling allows for multiple heterogeneous features in one vector. We introduce three types of acoustic vocabulary (AV) with mixture of token unigram, bigram, and trigram:

a) AV1: 32 broad class phonemes as unigram, selected from 12 languages, also referred to as P-ASM as detailed in (Ma *et al.*, 2005)
b) AV2: AV1 augmented by 32×32 bigrams of AV1, amounting to 1,056 tokens
c) AV3: AV2 augmented by $32 \times 32 \times 32$ trigrams of AV1, amounting to 33,824 tokens

	AV1	AV2	AV3
ER %	46.1	32.8	28.3

Table 1. Effect of acoustic vocabulary (KNC)

We carry out experiments with KNC classifier of 4,800 centroids. Applying *k*-nearest-neighboring rule, *k* is empirically set to 3. The error rates are reported in Table 1 for the experiments over the three AV types. It is found that high-order token *n*-grams improve LID performance. This reaffirms many previous findings that *n*-gram phonotactics serves as a valuable cue in LID.

4.2 Effect of Model Size

As discussed in KNC, one would expect to improve the phonotactic model by using more centroids. Let's examine how the number of centroid vectors *M* affects the performance of KNC. We set the acoustic system size *Y* to 128, *k*-nearest to 3, and only use token bigrams in the *bag-of-sounds* vector. In Table 2, it is not surprising to find that the performance improves as *M* increases. However, it is not practical to have large *M* because $L' = L \times M$ comparisons need to take place in each test trial.

#*M*	1,200	2,400	4,800	12,000
ER %	17.0	15.7	15.4	14.8

Table 2. Effect of number of centroids (KNC)

To reduce computation, MMC attempts to use less number of mixtures *M* to represent the phonotactic space. With the smoothing effect of the mixture model, we expect to use less computation to achieve similar performance as KNC. In the experiment reported in Table 3, we find that MMC (*M*=1,024) achieves 14.9% error rate, which almost equalizes the best result in the KNC experiment (*M*=12,000) with much less computation.

#*M*	4	16	64	256	1,024
ER %	29.6	26.4	19.7	16.0	14.9

Table 3. Effect of number of mixtures (MMC)

4.3 Discussion

The *bag-of-sounds* approach has achieved equal success in both 1996 and 2003 NIST LRE databases. As more results are published on the 1996 NIST LRE database, we choose it as the platform of comparison. In Table 4, we report the performance across different approaches in terms of error rate for a quick comparison. MMC presents a 12.4% ER reduction over the best reported result[4] (Torres-Carrasquillo *et al.*, 2002).

It is interesting to note that the *bag-of-sounds* classifier outperforms its P-PRLM counterpart by a wide margin (14.9% vs 22.0%). This is attributed to the global phonotactic features in λ_l^{LM}. The performance gain in (Torres-Carrasquillo *et al.*, 2002; Singer *et al.*, 2003) was obtained mainly by fusing scores from several classifiers, namely GMM, P-PRLM and SVM, to benefit from both acoustic and language model scores. Noting that the *bag-of-sounds* classifier in this work solely relies on the LM score, it is believed that fusing with scores from other classifiers will further boost the LID performance.

	ER %
P-PRLM[5]	22.0
P-PRLM + GMM acoustic[5]	19.5
P-PRLM + GMM acoustic + GMM tokenizer[5]	17.0
Bag-of-sounds classifier (MMC)	14.9

Table 4. Benchmark of different approaches

Besides the error rate reduction, the *bag-of-sounds* approach also simplifies the on-line computing procedure over its P-PRLM counterpart. It would be interesting to estimate the on-line computational need of MMC. The cost incurred has two main components: 1) the construction of the

[4] Previous results are also reported in DCF, DET, and equal error rate (EER). Comprehensive benchmarking for *bag-of-sounds* phonotactic LM will be reported soon.

[5] Results extracted from (Torres-Carrasquillo *et al.*, 2002)

pseudo document vector, as done via Eq.(5); 2) $L' = L \times M$ vector comparisons. The computing cost is estimated to be $\mathcal{O}(Q^2)$ per test trial (Bellegarda, 2000). For typical values of Q, this amounts to less than 0.05 Mflops. While this is more expensive than the usual table look-up in conventional *n*-gram LM, the performance improvement is able to justify the relatively modest computing overhead.

5 Conclusion

We have proposed a phonotactic LM approach to LID problem. The concept of *bag-of-sounds* is introduced, for the first time, to model phonotactics present in a spoken language over a larger context. With *bag-of-sounds* phonotactic LM, a spoken document can be treated as a text-like document of acoustic tokens. This way, the well-established LSA technique can be readily applied. This novel approach not only suggests a paradigm shift in LID, but also brings 12.4% error rate reduction over one of the best reported results on the 1996 NIST LRE data. It has proven to be very successful.

We would like to extend this approach to other spoken document categorization tasks. In monolingual spoken document categorization, we suggest that the semantic domain can be characterized by latent phonotactic features. Thus it is straightforward to extend the proposed *bag-of-sounds* framework to spoken document categorization.

Acknowledgement

The authors are grateful to Dr. Alvin F. Martin of the NIST Speech Group for his advice when preparing the 1996 NIST LRE experiments, to Dr G. M. White and Ms Y. Chen of Institute for Infocomm Research for insightful discussions.

References

Jerome R. Bellegarda. 2000. *Exploiting latent semantic information in statistical language modeling*, In Proc. of the IEEE, 88(8):1279-1296.

M. W. Berry, S.T. Dumais and G.W. O'Brien. 1995. *Using Linear Algebra for intelligent information retrieval*, SIAM Review, 37(4):573-595.

William B. Cavnar, and John M. Trenkle. 1994. *N-Gram-Based Text Categorization*, In Proc. of 3rd Annual Symposium on Document Analysis and Information Retrieval, pp. 161-169.

Jennifer Chu-Carroll, and Bob Carpenter. 1999. *Vector-based Natural Language Call Routing*, Computational Linguistics, 25(3):361-388.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, 1990, *Indexing by latent semantic analysis*, *Journal of the American Society for Informatin Science*, 41(6):391-407

Richard O. Duda and Peter E. Hart. 1973. *Pattern Classification and scene analysis*. John Wiley & Sons

James L. Hieronymus. 1994. *ASCII Phonetic Symbols for the World's Languages: Worldbet*. Technical Report AT&T Bell Labs.

Spark Jones, K. 1972. *A statistical interpretation of term specificity and its application in retrieval*, Journal of Documentation, 28:11-20

Bin Ma, Haizhou Li and Chin-Hui Lee, 2005. *An Acoustic Segment Modeling Approach to Automatic Language Identification*, submitted to Interspeech 2005

Yeshwant K. Muthusamy, Neena Jain, and Ronald A. Cole. 1994. *Perceptual benchmarks for automatic language identification*, In Proc. of ICASSP

Corinna Ng , Ross Wilkinson , Justin Zobel, 2000. *Experiments in spoken document retrieval using phoneme n-grams*, Speech Communication, 32(1-2):61-77

G. Salton, 1971. T*he SMART Retrieval System*, Prentice-Hall, Englewood Cliffs, NJ, 1971

E. Singer, P.A. Torres-Carrasquillo, T.P. Gleason, W.M. Campbell and D.A. Reynolds. 2003. *Acoustic, Phonetic and Discriminative Approaches to Automatic language recognition*, In Proc. of Eurospeech

Masahide Sugiyama. 1991. *Automatic language recognition using acoustic features*, In Proc. of ICASSP.

Pedro A. Torres-Carrasquillo, Douglas A. Reynolds, and J.R. Deller. Jr. 2002. *Language identification using Gaussian Mixture model tokenization*, in Proc. of ICASSP.

Yonghong Yan, and Etienne Barnard. 1995. *An approach to automatic language identification based on language dependent phone recognition*, In Proc. of ICASSP.

George K. Zipf. 1949. *Human Behavior and the Principal of Least effort, an introduction to human ecology*. Addison-Wesley, Reading, Mass.

Marc A. Zissman. 1996. *Comparison of four approaches to automatic language identification of telephone speech*, IEEE Trans. on Speech and Audio Processing, 4(1):31-44.

Reading Level Assessment Using Support Vector Machines and Statistical Language Models

Sarah E. Schwarm
Dept. of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350
sarahs@cs.washington.edu

Mari Ostendorf
Dept. of Electrical Engineering
University of Washington
Seattle, WA 98195-2500
mo@ee.washington.edu

Abstract

Reading proficiency is a fundamental component of language competency. However, finding topical texts at an appropriate reading level for foreign and second language learners is a challenge for teachers. This task can be addressed with natural language processing technology to assess reading level. Existing measures of reading level are not well suited to this task, but previous work and our own pilot experiments have shown the benefit of using statistical language models. In this paper, we also use support vector machines to combine features from traditional reading level measures, statistical language models, and other language processing tools to produce a better method of assessing reading level.

1 Introduction

The U.S. educational system is faced with the challenging task of educating growing numbers of students for whom English is a second language (U.S. Dept. of Education, 2003). In the 2001-2002 school year, Washington state had 72,215 students (7.2% of all students) in state programs for Limited English Proficient (LEP) students (Bylsma et al., 2003). In the same year, one quarter of all public school students in California and one in seven students in Texas were classified as LEP (U.S. Dept. of Education, 2004). Reading is a critical part of language and educational development, but finding appropriate reading material for LEP students is often difficult. To meet the needs of their students, bilingual education instructors seek out "high interest level" texts at low reading levels, e.g. texts at a first or second grade reading level that support the fifth grade science curriculum. Teachers need to find material at a variety of levels, since students need different texts to read independently and with help from the teacher. Finding reading materials that fulfill these requirements is difficult and time-consuming, and teachers are often forced to rewrite texts themselves to suit the varied needs of their students.

Natural language processing (NLP) technology is an ideal resource for automating the task of selecting appropriate reading material for bilingual students. Information retrieval systems successfully find topical materials and even answer complex queries in text databases and on the World Wide Web. However, an effective automated way to assess the reading level of the retrieved text is still needed. In this work, we develop a method of reading level assessment that uses support vector machines (SVMs) to combine features from statistical language models (LMs), parse trees, and other traditional features used in reading level assessment.

The results presented here on reading level assessment are part of a larger project to develop teacher-support tools for bilingual education instructors. The larger project will include a text simplification system, adapting paraphrasing and summarization techniques. Coupled with an information retrieval system, these tools will be used to select and simplify reading material in multiple languages for use by language learners. In addition to students in bilingual education, these tools will also be useful for those with reading-related learning disabilities and adult literacy students. In both of these situations, as in the bilingual education case, the student's reading level does not match his/her intellectual level and interests.

The remainder of the paper is organized as follows. Section 2 describes related work on reading level assessment. Section 3 describes the corpora used in our work. In Section 4 we present our approach to the task, and Section 5 contains experimental results. Section 6 provides a summary and description of future work.

2 Reading Level Assessment

This section highlights examples and features of some commonly used measures of reading level and discusses current research on the topic of reading level assessment using NLP techniques.

Many traditional methods of reading level assessment focus on simple approximations of syntactic complexity such as sentence length. The widely-used Flesch-Kincaid Grade Level index is based on the average number of syllables per word and the average sentence length in a passage of text (Kincaid et al., 1975) (as cited in (Collins-Thompson and Callan, 2004)). Similarly, the Gunning Fog index is based on the average number of words per sentence and the percentage of words with three or more syllables (Gunning, 1952). These methods are quick and easy to calculate but have drawbacks: sentence length is not an accurate measure of syntactic complexity, and syllable count does not necessarily indicate the difficulty of a word. Additionally, a student may be familiar with a few complex words (e.g. dinosaur names) but unable to understand complex syntactic constructions.

Other measures of readability focus on semantics, which is usually approximated by word frequency with respect to a reference list or corpus. The Dale-Chall formula uses a combination of average sentence length and percentage of words not on a list of 3000 "easy" words (Chall and Dale, 1995). The Lexile framework combines measures of semantics, represented by word frequency counts, and syntax, represented by sentence length (Stenner, 1996). These measures are inadequate for our task; in many cases, teachers want materials with more difficult, topic-specific words but simple structure. Measures of reading level based on word lists do not capture this information.

In addition to the traditional reading level metrics, researchers at Carnegie Mellon University have applied probabilistic language modeling techniques to this task. Si and Callan (2001) conducted preliminary work to classify science web pages using unigram models. More recently, Collins-Thompson and Callan manually collected a corpus of web pages ranked by grade level and observed that vocabulary words are not distributed evenly across grade levels. They developed a "smoothed unigram" classifier to better capture the variance in word usage across grade levels (Collins-Thompson and Callan, 2004). On web text, their classifier outperformed several other measures of semantic difficulty: the fraction of unknown words in the text, the number of distinct types per 100 token passage, the mean log frequency of the text relative to a large corpus, and the Flesch-Kincaid measure. The traditional measures performed better on some commercial corpora, but these corpora were calibrated using similar measures, so this is not a fair comparison. More importantly, the smoothed unigram measure worked better on the web corpus, especially on short passages. The smoothed unigram classifier is also more generalizable, since it can be trained on any collection of data. Traditional measures such as Dale-Chall and Lexile are based on static word lists.

Although the smoothed unigram classifier outperforms other vocabulary-based semantic measures, it does not capture syntactic information. We believe that higher order n-gram models or class n-gram models can achieve better performance by capturing both semantic and syntactic information. This is particularly important for the tasks we are interested in, when the vocabulary (i.e. topic) and grade level are not necessarily well-matched.

3 Corpora

Our work is currently focused on a corpus obtained from Weekly Reader, an educational newspaper with versions targeted at different grade levels (Weekly Reader, 2004). These data include a variety of labeled non-fiction topics, including science, history, and current events. Our corpus consists of articles from the second, third, fourth, and fifth grade edi-

Grade	Num Articles	Num Words
2	351	71.5k
3	589	444k
4	766	927k
5	691	1M

Table 1: Distribution of articles and words in the Weekly Reader corpus.

Corpus	Num Articles	Num Words
Britannica	115	277k
B. Elementary	115	74k
CNN	111	51k
CNN Abridged	111	37k

Table 2: Distribution of articles and words in the Britannica and CNN corpora.

tions of the newspaper. We design classifiers to distinguish each of these four categories. This corpus contains just under 2400 articles, distributed as shown in Table 1.

Additionally, we have two corpora consisting of articles for adults and corresponding simplified versions for children or other language learners. Barzilay and Elhadad (2003) have allowed us to use their corpus from Encyclopedia Britannica, which contains articles from the full version of the encyclopedia and corresponding articles from Britannica Elementary, a new version targeted at children. The Western/Pacific Literacy Network's (2004) web site has an archive of CNN news stories and abridged versions which we have also received permission to use. Although these corpora do not provide an explicit grade-level ranking for each article, broad categories are distinguished. We use these data as a supplement to the Weekly Reader corpus for learning models to distinguish broad reading level classes than can serve to provide features for more detailed classification. Table 2 shows the size of the supplemental corpora.

4 Approach

Existing reading level measures are inadequate due to their reliance on vocabulary lists and/or a superficial representation of syntax. Our approach uses n-gram language models as a low-cost automatic approximation of both syntactic and semantic analysis. Statistical language models (LMs) are used successfully in this way in other areas of NLP such as speech recognition and machine translation. We also use a standard statistical parser (Charniak, 2000) to provide syntactic analysis.

In practice, a teacher is likely to be looking for texts at a particular level rather than classifying a group of texts into a variety of categories. Thus we construct one classifier per category which decides whether a document belongs in that category or not, rather than constructing a classifier which ranks documents into different categories relative to each other.

4.1 Statistical Language Models

Statistical LMs predict the probability that a particular word sequence will occur. The most commonly used statistical language model is the n-gram model, which assumes that the word sequence is an $(n-1)$th order Markov process. For example, for the common trigram model where $n = 3$, the probability of sequence w is:

$$P(w) = P(w_1)P(w_2|w_1)\prod_{i=3}^{m} P(w_i|w_{i-1}, w_{i-2}). \tag{1}$$

The parameters of the model are estimated using a maximum likelihood estimate based on the observed frequency in a training corpus and smoothed using modified Kneser-Ney smoothing (Chen and Goodman, 1999). We used the SRI Language Modeling Toolkit (Stolcke, 2002) for language model training.

Our first set of classifiers consists of one n-gram language model per class c in the set of possible classes C. For each text document t, we can calculate the likelihood ratio between the probability given by the model for class c and the probabilities given by the other models for the other classes:

$$LR = \frac{P(t|c)P(c)}{\sum_{c' \neq c} P(t|c')P(c')} \tag{2}$$

where we assume uniform prior probabilities $P(c)$. The resulting value can be compared to an empirically chosen threshold to determine if the document is in class c or not. For each class c, a language model is estimated from a corpus of training texts.

In addition to using the likelihood ratio for classification, we can use scores from language models as features in another classifier (e.g. an SVM). For example, perplexity (PP) is an information-theoretic measure often used to assess language models:

$$PP = 2^{H(t|c)}, \tag{3}$$

where $H(t|c)$ is the entropy relative to class c of a length m word sequence $t = w_1, ..., w_m$, defined as

$$H(t|c) = -\frac{1}{m} \log_2 P(t|c). \tag{4}$$

Low perplexity indicates a better match between the test data and the model, corresponding to a higher probability $P(t|c)$. Perplexity scores are used as features in the SVM model described in Section 4.3. The likelihood ratio described above could also be used as a feature, but we achieved better results using perplexity.

4.2 Feature Selection

Feature selection is a common part of classifier design for many classification problems; however, there are mixed results in the literature on feature selection for text classification tasks. In Collins-Thompson and Callan's work (2004) on readability assessment, LM smoothing techniques are more effective than other forms of explicit feature selection. However, feature selection proves to be important in other text classification work, e.g. Lee and Myaeng's (2002) genre and subject detection work and Boulis and Ostendorf's (2005) work on feature selection for topic classification.

For our LM classifiers, we followed Boulis and Ostendorf's (2005) approach for feature selection and ranked words by their ability to discriminate between classes. Given $P(c|w)$, the probability of class c given word w, estimated empirically from the training set, we sorted words based on their information gain (IG). Information gain measures the difference in entropy when w is and is not included as a feature.

$$\begin{aligned} IG(w) = & - \sum_{c \in C} P(c) \log P(c) \\ & + P(w) \sum_{c \in C} P(c|w) \log P(c|w) \\ & + P(\bar{w}) \sum_{c \in C} P(c|\bar{w}) \log P(c|\bar{w}). \end{aligned} \tag{5}$$

The most discriminative words are selected as features by plotting the sorted IG values and keeping only those words below the "knee" in the curve, as determined by manual inspection of the graph. In an early experiment, we replaced all remaining words with a single "unknown" tag. This did not result in an effective classifier, so in later experiments the remaining words were replaced with a small set of general tags. Motivated by our goal of representing syntax, we used part-of-speech (POS) tags as labeled by a maximum entropy tagger (Ratnaparkhi, 1996). These tags allow the model to represent patterns in the text at a higher level than that of individual words, using sequences of POS tags to capture rough syntactic information. The resulting vocabulary consisted of 276 words and 56 POS tags.

4.3 Support Vector Machines

Support vector machines (SVMs) are a machine learning technique used in a variety of text classification problems. SVMs are based on the principle of structural risk minimization. Viewing the data as points in a high-dimensional feature space, the goal is to fit a hyperplane between the positive and negative examples so as to maximize the distance between the data points and the plane. SVMs were introduced by Vapnik (1995) and were popularized in the area of text classification by Joachims (1998a).

The unit of classification in this work is a single article. Our SVM classifiers for reading level use the following features:

- Average sentence length
- Average number of syllables per word
- Flesch-Kincaid score
- 6 out-of-vocabulary (OOV) rate scores.
- Parse features (per sentence):
  - Average parse tree height
  - Average number of noun phrases
  - Average number of verb phrases
  - Average number of "SBAR"s.[1]
- 12 language model perplexity scores

The OOV scores are relative to the most common 100, 200 and 500 words in the lowest grade level

[1]SBAR is defined in the Penn Treebank tag set as a "clause introduced by a (possibly empty) subordinating conjunction." It is an indicator of sentence complexity.

(grade 2) [2]. For each article, we calculated the percentage of a) all word instances (tokens) and b) all unique words (types) not on these lists, resulting in three token OOV rate features and three type OOV rate features per article.

The parse features are generated using the Charniak parser (Charniak, 2000) trained on the standard Wall Street Journal Treebank corpus. We chose to use this standard data set as we do not have any domain-specific treebank data for training a parser. Although clearly there is a difference between news text for adults and news articles intended for children, inspection of some of the resulting parses showed good accuracy.

Ideally, the language model scores would be for LMs from domain-specific training data (i.e. more Weekly Reader data.) However, our corpus is limited and preliminary experiments in which the training data was split for LM and SVM training were unsuccessful due to the small size of the resulting data sets. Thus we made use of the Britannica and CNN articles to train models of three n-gram orders on "child" text and "adult" text. This resulted in 12 LM perplexity features per article based on trigram, bigram and unigram LMs trained on Britannica (adult), Britannica Elementary, CNN (adult) and CNN abridged text.

For training SVMs, we used the SVM^{light} toolkit developed by Joachims (1998b). Using development data, we selected the radial basis function kernel and tuned parameters using cross validation and grid search as described in (Hsu et al., 2003).

5 Experiments

5.1 Test Data and Evaluation Criteria

We divide the Weekly Reader corpus described in Section 3 into separate training, development, and test sets. The number of articles in each set is shown in Table 3. The development data is used as a test set for comparing classifiers, tuning parameters, etc, and the results presented in this section are based on the test set.

Grade	**Training**	**Dev/Test**
2	315	18
3	529	30
4	690	38
5	623	34

Table 3: Number of articles in the Weekly Reader corpus as divided into training, development and test sets. The dev and test sets are the same size and each consist of approximately 5% of the data for each grade level.

We present results in three different formats. For analyzing our binary classifiers, we use Detection Error Tradeoff (DET) curves and precision/recall measures. For comparison to other methods, e.g. Flesch-Kincaid and Lexile, which are not binary classifiers, we consider the percentage of articles which are misclassified by more than one grade level.

Detection Error Tradeoff curves show the tradeoff between misses and false alarms for different threshold values for the classifiers. "Misses" are positive examples of a class that are misclassified as negative examples; "false alarms" are negative examples misclassified as positive. DET curves have been used in other detection tasks in language processing, e.g. Martin et al. (1997). We use these curves to visualize the tradeoff between the two types of errors, and select the minimum cost operating point in order to get a threshold for precision and recall calculations. The minimum cost operating point depends on the relative costs of misses and false alarms; it is conceivable that one type of error might be more serious than the other. After consultation with teachers (future users of our system), we concluded that there are pros and cons to each side, so for the purpose of this analysis we weighted the two types of errors equally. In this work, the minimum cost operating point is selected by averaging the percentages of misses and false alarms at each point and choosing the point with the lowest average. Unless otherwise noted, errors reported are associated with these actual operating points, which may not lie on the convex hull of the DET curve.

Precision and recall are often used to assess information retrieval systems, and our task is similar. Precision indicates the percentage of the retrieved documents that are relevant, in this case the percentage of detected documents that match the target

[2]These lists are chosen from the full vocabulary independently of the feature selection for LMs described above.

grade level. Recall indicates the percentage of the total number of relevant documents in the data set that are retrieved, in this case the percentage of the total number of documents from the target level that are detected.

5.2 Language Model Classifier

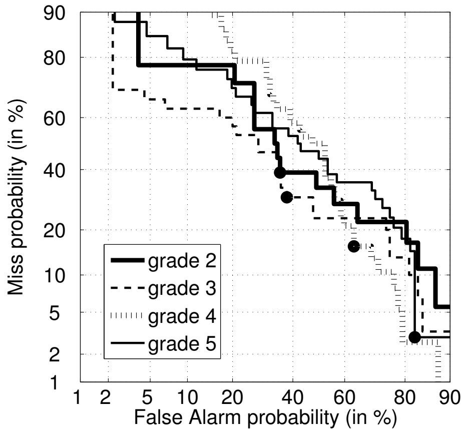


Figure 1: DET curves (test set) for classifiers based on trigram language models.

Figure 1 shows DET curves for the trigram LM-based classifiers. The minimum cost error rates for these classifiers, indicated by large dots in the plot, are in the range of 33-43%, with only one over 40%. The curves for bigram and unigram models have similar shapes, but the trigram models outperform the lower-order models. Error rates for the bigram models range from 37-45% and the unigram models have error rates in the 39-49% range, with all but one over 40%. Although our training corpus is small the feature selection described in Section 4.2 allows us to use these higher-order trigram models.

5.3 Support Vector Machine Classifier

By combining language model scores with other features in an SVM framework, we achieve our best results. Figures 2 and 3 show DET curves for this set of classifiers on the development set and test set, respectively. The grade 2 and 5 classifiers have the best performance, probably because grade 3 and 4 must be distinguished from other classes at both higher and lower levels. Using threshold values selected based on minimum cost on the development

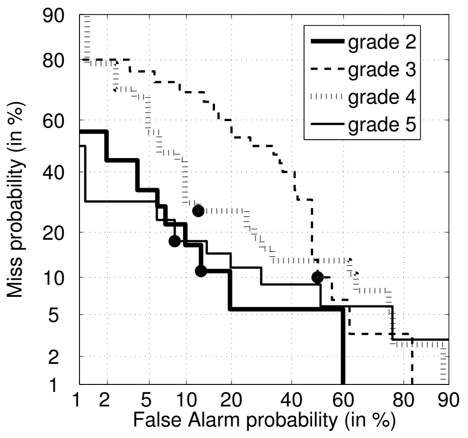


Figure 2: DET curves (development set) for SVM classifiers with LM features.

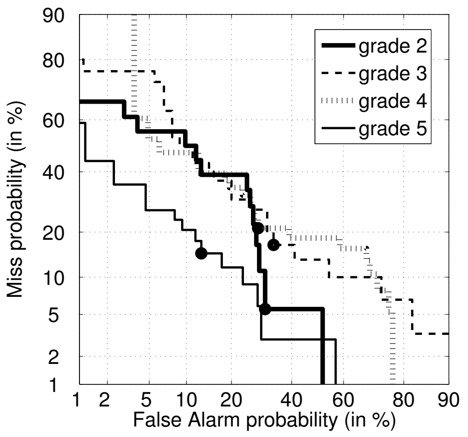


Figure 3: DET curves (test set) for SVM classifiers with LM features.

set, indicated by large dots on the plot, we calculated precision and recall on the test set. Results are presented in Table 4. The grade 3 classifier has high recall but relatively low precision; the grade 4 classifier does better on precision and reasonably well on recall. Since the minimum cost operating points do not correspond to the equal error rate (i.e. equal percentage of misses and false alarms) there is variation in the precision-recall tradeoff for the different grade level classifiers. For example, for class 3, the operating point corresponds to a high probability of false alarms and a lower probability of misses, which results in low precision and high recall. For operating points chosen on the convex hull of the DET curves, the equal error rate ranges from 12-25% for the dif-

Grade	Precision	Recall
2	38%	61%
3	38%	87%
4	70%	60%
5	75%	79%

Table 4: Precision and recall on test set for SVM-based classifiers.

Grade	Errors		
	Flesch-Kincaid	Lexile	SVM
2	78%	33%	5.5%
3	67%	27%	3.3%
4	74%	26%	13%
5	59%	24%	21%

Table 5: Percentage of articles which are misclassified by more than one grade level.

ferent grade levels.

We investigated the contribution of individual features to the overall performance of the SVM classifier and found that no features stood out as most important, and performance was degraded when any particular features were removed.

5.4 Comparison

We also compared error rates for the best performing SVM classifier with two traditional reading level measures, Flesch-Kincaid and Lexile. The Flesch-Kincaid Grade Level index is a commonly used measure of reading level based on the average number of syllables per word and average sentence length. The Flesch-Kincaid score for a document is intended to directly correspond with its grade level. We chose the Lexile measure as an example of a reading level classifier based on word lists.[3] Lexile scores do not correlate directly to numeric grade levels, however a mapping of ranges of Lexile scores to their corresponding grade levels is available on the Lexile web site (Lexile, 2005).

For each of these three classifiers, Table 5 shows the percentage of articles which are misclassified by more than one grade level. Flesch-Kincaid performs poorly, as expected since its only features are sentence length and average syllable count. Although this index is commonly used, perhaps due to its simplicity, it is not accurate enough for the intended application. Our SVM classifier also outperforms the Lexile metric. Lexile is a more general measure while our classifier is trained on this particular domain, so the better performance of our model is not entirely surprising. Importantly, however, our classifier is easily tuned to any corpus of interest.

To test our classifier on data outside the Weekly Reader corpus, we downloaded 10 randomly selected newspaper articles from the "Kidspost" edition of The Washington Post (2005). "Kidspost" is intended for grades 3-8. We found that our SVM classifier, trained on the Weekly Reader corpus, classified four of these articles as grade 4 and seven articles as grade 5 (with one overlap with grade 4). These results indicate that our classifier can generalize to other data sets. Since there was no training data corresponding to higher reading levels, the best performance we can expect for adult-level newspaper articles is for our classifiers to mark them as the highest grade level, which is indeed what happened for 10 randomly chosen articles from standard edition of The Washington Post.

6 Conclusions and Future Work

Statistical LMs were used to classify texts based on reading level, with trigram models being noticeably more accurate than bigrams and unigrams. Combining information from statistical LMs with other features using support vector machines provided the best results. Future work includes testing additional classifier features, e.g. parser likelihood scores and features obtained using a syntax-based language model such as Chelba and Jelinek (2000) or Roark (2001). Further experiments are planned on the generalizability of our classifier to text from other sources (e.g. newspaper articles, web pages); to accomplish this we will add higher level text as negative training data. We also plan to test these techniques on languages other than English, and incorporate them with an information retrieval system to create a tool that may be used by teachers to help select reading material for their students.

[3]Other classifiers such as Dale-Chall do not have automatic software available.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0326276. 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 National Science Foundation.

Thank you to Paul Heavenridge (Literacyworks), the Weekly Reader Corporation, Regina Barzilay (MIT) and Noemie Elhadad (Columbia University) for sharing their data and corpora.

References

R. Barzilay and N. Elhadad. Sentence alignment for monolingual comparable corpora. In *Proc. of EMNLP*, pages 25–32, 2003.

C. Boulis and M. Ostendorf. Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams. *Workshop on Feature Selection in Data Mining, in conjunction with SIAM conference on Data Mining*, 2005.

P. Bylsma, L. Ireland, and H. Malagon. *Educating English Language Learners in Washington State*. Office of the Superintendent of Public Instruction, Olympia, WA, 2003.

J.S. Chall and E. Dale. *Readability revisited: the new Dale-Chall readability formula*. Brookline Books, Cambridge, Mass., 1995.

E. Charniak. A maximum-entropy-inspired parser. In *Proc. of NAACL*, pages 132–139, 2000.

C. Chelba and F. Jelinek. Structured Language Modeling. *Computer Speech and Language*, 14(4):283-332, 2000.

S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–393, 1999.

K. Collins-Thompson and J. Callan. A language modeling approach to predicting reading difficulty. In *Proc. of HLT/NAACL*, pages 193–200, 2004.

R. Gunning. *The technique of clear writing*. McGraw-Hill, New York, 1952.

C.-W. Hsu et al. A practical guide to support vector classification. `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`, 2003. Accessed 11/2004.

T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proc. of the European Conference on Machine Learning*, pages 137–142, 1998a.

T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*. B. Schölkopf, C. Burges, A. Smola, eds. MIT Press, Cambridge, MA, 1998b.

J.P. Kincaid, Jr., R.P. Fishburne, R.L. Rodgers, and B.S. Chisson. Derivation of new readability formulas for Navy enlisted personnel. Research Branch Report 8-75, U.S. Naval Air Station, Memphis, 1975.

Y.-B. Lee and S.H. Myaeng. Text genre classification with genre-revealing and subject-revealing features. In *Proc. of SIGIR*, pages 145–150, 2002.

The Lexile framework for reading. `http://www.lexile.com`, 2005. Accessed April 15, 2005.

A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. *Proc. of Eurospeech*, v. 4, pp. 1895-1898, 1997.

A. Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proc. of EMNLP*, pages 133–141, 1996.

B. Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249-276, 2001.

L. Si and J.P. Callan. A statistical model for scientific readability. In *Proc. of CIKM*, pages 574–576, 2001.

A.J. Stenner. Measuring reading comprehension with the Lexile framework. Presented at the Fourth North American Conference on Adolescent/Adult Literacy, 1996.

A. Stolcke. SRILM - an extensible language modeling toolkit. *Proc. ICSLP*, v. 2, pp. 901-904, 2002.

U.S. Department of Education, National Center for Educational Statistics. The condition of education. `http://nces.ed.gov/programs/coe/2003/section1/indicator04.asp`, 2003. Accessed June 18, 2004.

U.S. Department of Education, National Center for Educational Statistics. NCES fast facts: Bilingual education/Limited English Proficient students. `http://nces.ed.gov/fastfacts/display.asp?id=96`, 2003. Accessed June 18, 2004.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

The Washington Post. `http://www.washingtonpost.com`, 2005. Accessed April 20, 2005.

Weekly Reader. `http://www.weeklyreader.com`, 2004. Accessed July, 2004.

Western/Pacific Literacy Network / Literacyworks. CNN SF learning resources. `http://literacynet.org/cnnsf/`, 2004. Accessed June 15, 2004.

Clause Restructuring for Statistical Machine Translation

Michael Collins
MIT CSAIL
mcollins@csail.mit.edu

Philipp Koehn
School of Informatics
University of Edinburgh
pkoehn@inf.ed.ac.uk

Ivona Kučerová
MIT Linguistics Department
kucerova@mit.edu

Abstract

We describe a method for incorporating syntactic information in statistical machine translation systems. The first step of the method is to parse the source language string that is being translated. The second step is to apply a series of transformations to the parse tree, effectively reordering the surface string on the source language side of the translation system. The goal of this step is to recover an underlying word order that is closer to the target language word-order than the original string. The reordering approach is applied as a pre-processing step in both the training and decoding phases of a phrase-based statistical MT system. We describe experiments on translation from German to English, showing an improvement from 25.2% Bleu score for a baseline system to 26.8% Bleu score for the system with reordering, a statistically significant improvement.

1 Introduction

Recent research on statistical machine translation (SMT) has lead to the development of *phrase-based* systems (Och et al., 1999; Marcu and Wong, 2002; Koehn et al., 2003). These methods go beyond the original IBM machine translation models (Brown et al., 1993), by allowing multi-word units ("phrases") in one language to be translated directly into phrases in another language. A number of empirical evaluations have suggested that phrase-based systems currently represent the state–of–the–art in statistical machine translation.

In spite of their success, a key limitation of phrase-based systems is that they make little or no direct use of syntactic information. It appears likely that syntactic information will be crucial in accurately modeling many phenomena during translation, for example systematic differences between the word order of different languages. For this reason there is currently a great deal of interest in methods which incorporate syntactic information within statistical machine translation systems (e.g., see (Alshawi, 1996; Wu, 1997; Yamada and Knight, 2001; Gildea, 2003; Melamed, 2004; Graehl and Knight, 2004; Och et al., 2004; Xia and McCord, 2004)).

In this paper we describe an approach for the use of syntactic information within phrase-based SMT systems. The approach constitutes a simple, direct method for the incorporation of syntactic information in a phrase–based system, which we will show leads to significant improvements in translation accuracy. The first step of the method is to parse the source language string that is being translated. The second step is to apply a series of transformations to the resulting parse tree, effectively reordering the surface string on the source language side of the translation system. The goal of this step is to recover an underlying word order that is closer to the target language word-order than the original string. Finally, we apply a phrase-based system to the reordered string to give a translation into the target language.

We describe experiments involving machine translation from German to English. As an illustrative example of our method, consider the following German sentence, together with a "translation" into English that follows the original word order:

Original sentence: Ich werde Ihnen die entsprechenden Anmerkungen aushaendigen, damit Sie das eventuell bei der Abstimmung uebernehmen koennen.

English translation: *I will to you the corresponding comments pass on, so that you them perhaps in the vote adopt can.*

The German word order in this case is substantially different from the word order that would be seen in English. As we will show later in this paper, translations of sentences of this type pose difficulties for phrase-based systems. In our approach we reorder the constituents in a parse of the German sentence to give the following word order, which is much closer to the target English word order (words which have been "moved" are underlined):

Reordered sentence: Ich werde <u>aushaendigen</u> Ihnen die entsprechenden Anmerkungen, damit Sie <u>koennen uebernehmen</u> das eventuell bei der Abstimmung.

English translation: *I will <u>pass on</u> to you the corresponding comments, so that you <u>can</u> <u>adopt</u> them perhaps in the vote.*

We applied our approach to translation from German to English in the Europarl corpus. Source language sentences are reordered in test data, and also in training data that is used by the underlying phrase-based system. Results using the method show an improvement from 25.2% Bleu score to 26.8% Bleu score (a statistically signifi cant improvement), using a phrase-based system (Koehn et al., 2003) which has been shown in the past to be a highly competitive SMT system.

2 Background

2.1 Previous Work

2.1.1 Research on Phrase-Based SMT

The original work on statistical machine translation was carried out by researchers at IBM (Brown et al., 1993). More recently, phrase-based models (Och et al., 1999; Marcu and Wong, 2002; Koehn et al., 2003) have been proposed as a highly successful alternative to the IBM models. Phrase-based models generalize the original IBM models by allowing multiple words in one language to correspond to multiple words in another language. For example, we might have a translation entry specifying that *I will* in English is a likely translation for *Ich werde* in German.

In this paper we use the phrase-based system of (Koehn et al., 2003) as our underlying model. This approach fi rst uses the original IBM models to derive word-to-word alignments in the corpus of example translations. Heuristics are then used to grow these alignments to encompass phrase-to-phrase pairs. The end result of the training process is a lexicon of phrase-to-phrase pairs, with associated costs or probabilities. In translation with the system, a beam search method with left-to-right search is used to fi nd a high scoring translation for an input sentence. At each stage of the search, one or more English words are added to the hypothesized string, and one or more consecutive German words are "absorbed" (i.e., marked as having already been translated—note that each word is absorbed at most once). Each step of this kind has a number of costs: for example, the log probability of the phrase-to-phrase correspondance involved, the log probability from a language model, and some "distortion" score indicating how likely it is for the proposed words in the English string to be aligned to the corresponding position in the German string.

2.1.2 Research on Syntax-Based SMT

A number of researchers (Alshawi, 1996; Wu, 1997; Yamada and Knight, 2001; Gildea, 2003; Melamed, 2004; Graehl and Knight, 2004; Galley et al., 2004) have proposed models where the translation process involves syntactic representations of the source and/or target languages. One class of approaches make use of "bitext" grammars which simultaneously parse both the source and target languages. Another class of approaches make use of syntactic information in the target language alone, effectively transforming the translation problem into a parsing problem. Note that these models have radically different structures and parameterizations from phrase–based models for SMT. As yet, these systems have not shown signifi cant gains in accuracy in comparison to phrase-based systems.

Reranking methods have also been proposed as a method for using syntactic information (Koehn and Knight, 2003; Och et al., 2004; Shen et al., 2004). In these approaches a baseline system is used to generate N-best output. Syntactic features are then used in a second model that reranks the N-best lists, in an attempt to improve over the baseline approach. (Koehn and Knight, 2003) apply a reranking approach to the sub-task of noun-phrase translation. (Och et al., 2004; Shen et al., 2004) describe the use of syntactic features in reranking the output of a full translation system, but the syntactic features give very small gains: for example the majority of the gain in performance in the experiments in (Och et al., 2004) was due to the addition of IBM Model 1 translation probabilities, a non-syntactic feature.

An alternative use of syntactic information is to employ an existing statistical parsing model as a language model within an SMT system. See (Charniak et al., 2003) for an approach of this form, which shows improvements in accuracy over a baseline system.

2.1.3 Research on Preprocessing Approaches

Our approach involves a preprocessing step, where sentences in the language being translated are modifi ed before being passed to an existing phrase-based translation system. A number of other researchers (Berger et al., 1996; Niessen and Ney, 2004; Xia and McCord, 2004) have described previous work on preprocessing methods. (Berger et al., 1996) describe an approach that targets translation of French phrases of the form *NOUN de NOUN* (e.g., *conflit d'intérêt*). This was a relatively limited study, concentrating on this one syntactic phenomenon which involves relatively local transformations (a parser was not required in this study). (Niessen and Ney, 2004) describe a method that combines morphologically–split verbs in German, and also reorders questions in English and German. Our method goes beyond this approach in several respects, for example considering phenomena such as declarative (non-question) clauses, subordinate clauses, negation, and so on.

(Xia and McCord, 2004) describe an approach for translation from French to English, where reordering rules are acquired automatically. The reordering rules in their approach operate at the level of context-free rules in the parse tree. Our method differs from that of (Xia and McCord, 2004) in a couple of important respects. First, we are considering German, which arguably has more challenging word order phenonema than French. German has relatively free word order, in contrast to both English and French: for example, there is considerable flexibility in terms of which phrases can appear in the first position in a clause. Second, Xia et. al's (2004) use of reordering rules stated at the context-free level differs from ours. As one example, in our approach we use a single transformation that moves an infinitival verb to the first position in a verb phrase. Xia et. al's approach would require learning of a different rule transformation for every production of the form `VP => ....` In practice the German parser that we are using creates relatively "flat" structures at the VP and clause levels, leading to a huge number of context-free rules (the flatness is one consequence of the relatively free word order seen within VP's and clauses in German). There are clearly some advantages to learning reordering rules automatically, as in Xia et. al's approach. However, we note that our approach involves a handful of linguistically–motivated transformations and achieves comparable improvements (albeit on a different language pair) to Xia et. al's method, which in contrast involves over 56,000 transformations.

```
S PPER-SB  Ich
  VAFIN-HD  werde
  VP PPER-DA  Ihnen
     NP-OA ART   die
           ADJA  entsprechenden
           NN    Anmerkungen
     VVINF-HD    aushaendigen
     ,  ,
     S KOUS     damit
       PPER-SB  Sie
       VP PDS-OA  das
          ADJD eventuell
          PP APPR bei
             ART  der
             NN   Abstimmung
          VVINF-HD  uebernehmen
       VMFIN-HD  koennen
```

Figure 1: An example parse tree. Key to non-terminals: `PPER` = personal pronoun; `VAFIN` = finite verb; `VVINF` = infinitival verb; `KOUS` = complementizer; `APPR` = preposition; `ART` = article; `ADJA` = adjective; `ADJD` = adverb; `-SB` = subject; `-HD` = head of a phrase; `-DA` = dative object; `-OA` = accusative object.

2.2 German Clause Structure

In this section we give a brief description of the syntactic structure of German clauses. The characteristics we describe motivate the reordering rules described later in the paper.

Figure 1 gives an example parse tree for a German sentence. This sentence contains two clauses:

Clause 1: Ich/*I* werde/*will* Ihnen/*to_you* die/*the* entsprechenden/*corresponding* Anmerkungen/*comments* aushaendigen/*pass_on*

Clause 2: damit/*so_that* Sie/*you* das/*them* eventuell/*perhaps* bei/*in* der/*the* Abstimmung/*vote* uebernehmen/*adopt* koennen/*can*

These two clauses illustrate a number of syntactic phenomena in German which lead to quite different word order from English:

Position of finite verbs. In Clause 1, which is a matrix clause, the finite verb *werde* is in the second position in the clause. Finite verbs appear rigidly in 2nd position in matrix clauses. In contrast, in subordinate clauses, such as Clause 2, the finite verb comes last in the clause. For example, note that *koennen* is a finite verb which is the final element of Clause 2.

Position of infinitival verbs. In German, infinitival verbs are final within their associated verb phrase. For example, returning to Figure 1, notice that *aushaendigen* is the last element in its verb phrase, and that *uebernehmen* is the final element of its verb phrase in the figure.

Relatively flexible word ordering. German has substantially freer word order than English. In particular, note that while the verb comes second in matrix clauses, essentially any element can be in the first position. For example, in Clause 1, while the subject *Ich* is seen in the first position, potentially any of the other constituents (e.g., *Ihnen*) could also appear in this position. Note that this often leads to the subject following the finite verb, something which happens very rarely in English.

There are many other phenomena which lead to differing word order between German and English. Two others that we focus on in this paper are negation (the differing placement of items such as *not* in English and *nicht* in German), and also verb-particle constructions. We describe our treatment of these phenomena later in this paper.

2.3 Reordering with Phrase-Based SMT

We have seen in the last section that German syntax has several characteristics that lead to significantly different word order from that of English. We now describe how these characteristics can lead to difficulties for phrase–based translation systems when applied to German to English translation.

Typically, reordering models in phrase-based systems are based solely on movement distance. In particular, at each point in decoding a "cost" is associated with skipping over 1 or more German words. For example, assume that in translating

> Ich werde Ihnen die entsprechenden Anmerkungen aushaendigen.

we have reached a state where "Ich" and "werde" have been translated into "I will" in English. A potential decoding decision at this point is to add the phrase "pass on" to the English hypothesis, at the same time absorbing "aushaendigen" from the German string. The cost of this decoding step will involve a number of factors, including a cost of skipping over a phrase of length 4 (i.e., *Ihnen die entsprechenden Anmerkungen*) in the German string.

The ability to penalise "skips" of this type, and the potential to model multi-word phrases, are essentially the main strategies that the phrase-based system is able to employ when modeling differing word-order across different languages. In practice, when training the parameters of an SMT system, for example using the discriminative methods of (Och, 2003), the cost for skips of this kind is typically set to a very high value. In experiments with the system of (Koehn et al., 2003) we have found that in practice a large number of complete translations are completely monotonic (i.e., have 0 skips), suggesting that the system has difficulty learning exactly what points in the translation should allow reordering. In summary, phrase-based systems have relatively limited potential to model word-order differences between different languages.

The reordering stage described in this paper attempts to modify the source language (e.g., German) in such a way that its word order is very similar to that seen in the target language (e.g., English). In an ideal approach, the resulting translation problem that is passed on to the phrase-based system will be solvable using a completely monotonic translation, without any skips, and without requiring extremely long phrases to be translated (for example a phrasal translation corresponding to *Ihnen die entsprechenden Anmerkungen aushaendigen*).

Note than an additional benefit of the reordering phase is that it may bring together groups of words in German which have a natural correspondance to phrases in English, but were unseen or rare in the original German text. For example, in the previous example, we might derive a correspondance between *werde aushaendigen* and *will pass on* that was not possible before reordering. Another example concerns verb-particle constructions, for example in

> Wir machen die Tuer auf

machen and *auf* form a verb-particle construction. The reordering stage moves *auf* to precede *machen*, allowing a phrasal entry that "auf machen" is translated to *to open* in English. Without the reordering, the particle can be arbitrarily far from the verb that it modifies, and there is a danger in this example of translating *machen* as *to make*, the natural translation when no particle is present.

Original sentence: Ich werde Ihnen die entsprechenden Anmerkungen aushaendigen, damit Sie das eventuell bei der Abstimmung uebernehmen koennen. (*I will to you the corresponding comments pass on, so that you them perhaps in the vote adopt can.*)

Reordered sentence: Ich werde aushaendigen Ihnen die entsprechenden Anmerkungen, damit Sie koennen uebernehmen das eventuell bei der Abstimmung.
(*I will pass on to you the corresponding comments, so that you can adopt them perhaps in the vote.*)

Figure 2: An example of the reordering process, showing the original German sentence and the sentence after reordering.

3 Clause Restructuring

We now describe the method we use for reordering German sentences. As a first step in the reordering process, we parse the sentence using the parser described in (Dubey and Keller, 2003). The second step is to apply a sequence of rules that reorder the German sentence depending on the parse tree structure. See Figure 2 for an example German sentence before and after the reordering step.

In the reordering phase, each of the following six restructuring steps were applied to a German parse tree, in sequence (see table 1 also, for examples of the reordering steps):

[1] Verb initial In any verb phrase (i.e., phrase with label `VP-...`) find the head of the phrase (i.e., the child with label `-HD`) and move it into the initial position within the verb phrase. For example, in the parse tree in Figure 1, *aushaendigen* would be moved to precede *Ihnen* in the first verb phrase (`VP-OC`), and *uebernehmen* would be moved to precede *das* in the second `VP-OC`. The subordinate clause would have the following structure after this transformation:

```
S-MO KOUS-CP  damit
     PPER-SB  Sie
     VP-OC VVINF-HD  uebernehmen
           PDS-OA  das
           ADJD-MO  eventuell
           PP-MO APPR-DA  bei
                 ART-DA  der
                 NN-NK  Abstimmung
     VMFIN-HD  koennen
```

[2] Verb 2nd In any subordinate clause labelled `S-...`, with a complementizer `KOUS`, `PREL`, `PWS` or `PWAV`, find the head of the clause, and move it to directly follow the complementizer.

For example, in the subordinate clause in Figure 1, the head of the clause *koennen* would be moved to follow the complementizer *damit*, giving the following structure:

```
S-MO KOUS-CP  damit
     VMFIN-HD  koennen
     PPER-SB  Sie
     VP-OC VVINF-HD  uebernehmen
           PDS-OA  das
           ADJD-MO  eventuell
           PP-MO APPR-DA  bei
                 ART-DA  der
                 NN-NK  Abstimmung
```

[3] Move Subject For any clause (i.e., phrase with label `S...`), move the subject to directly precede the head. We define the subject to be the left-most child of the clause with label `...-SB` or `PPER-EP`, and the head to be the leftmost child with label `...-HD`.

For example, in the subordinate clause in Figure 1, the subject *Sie* would be moved to precede *koennen*, giving the following structure:

```
S-MO KOUS-CP  damit
     PPER-SB  Sie
     VMFIN-HD  koennen
     VP-OC VVINF-HD  uebernehmen
           PDS-OA  das
           ADJD-MO  eventuell
           PP-MO APPR-DA  bei
                 ART-DA  der
                 NN-NK  Abstimmung
```

[4] Particles In verb particle constructions, move the particle to immediately precede the verb. More specifically, if a finite verb (i.e., verb tagged as `VVFIN`) and a particle (i.e., word tagged as `PTKVZ`) are found in the same clause, move the particle to precede the verb.

As one example, the following clause contains both a verb (*forden*) as well as a particle (*auf*):

```
S PPER-SB   Wir
  VVFIN-HD  fordern
  NP-OA ART das
        NN  Praesidium
  PTKVZ-SVP auf
```

After the transformation, the clause is altered to:

```
S PPER-SB   Wir
  PTKVZ-SVP auf
  VVFIN-HD  fordern
  NP-OA ART das
        NN  Praesidium
```

Transformation	Example	
Verb Initial	Before:	Ich werde Ihnen die entsprechenden Anmerkungen **aushaendigen**, . . .
	After:	Ich werde **aushaendigen** Ihnen die entsprechenden Anmerkungen, . . .
		I shall be passing on to you some comments, . . .
Verb 2nd	Before:	. . . damit Sie uebernehmen das eventuell bei der Abstimmung **koennen**.
	After:	. . . damit **koennen** Sie uebernehmen das eventuell bei der Abstimmung .
		. . . so that could you adopt this perhaps in the voting.
Move Subject	Before:	. . . damit koennen **Sie** uebernehmen das eventuell bei der Abstimmung.
	After:	. . . damit **Sie** koennen uebernehmen das eventuell bei der Abstimmung .
		. . . so that you could adopt this perhaps in the voting.
Particles	Before:	Wir fordern das Praesidium **auf**, . . .
	After:	Wir **auf** fordern das Praesidium, . . .
		We ask the Bureau, . . .
Infinitives	Before:	Ich werde der Sache **nachgehen** dann, . . .
	After:	Ich werde **nachgehen** der Sache dann, . . .
		I will look into the matter then, . . .
Negation	Before:	Wir konnten einreichen es **nicht** mehr rechtzeitig, . . .
	After:	Wir konnten **nicht** einreichen es mehr rechtzeitig, . . .
		We could not hand it in in time, . . .

Table 1: Examples for each of the reordering steps. In each case the item that is moved is underlined.

[5] Infinitives In some cases, infinitival verbs are still not in the correct position after transformations [1]–[4]. For this reason we add a second step that involves infinitives. First, we remove all internal `VP` nodes within the parse tree. Second, for any clause (i.e., phrase labeled `S...`), if the clause dominates both a finite and infinitival verb, and there is an argument (i.e., a subject, or an object) between the two verbs, then the infinitive is moved to directly follow the finite verb.

As an example, the following clause contains an infinitival (*einreichen*) that is separated from a finite verb *konnten* by the direct object *es*:

```
S PPER-SB  Wir
  VMFIN-HD  konnten
  PPER-OA  es
  PTKNEG-NG  nicht
  VP-OC VVINF-HD  einreichen
        AP-MO ADV-MO  mehr
              ADJD-HD  rechtzeitig
```

The transformation removes the `VP-OC`, and moves the infinitive, giving:

```
S PPER-SB  Wir
  VMFIN-HD  konnten
  VVINF-HD  einreichen
  PPER-OA  es
  PTKNEG-NG  nicht
  AP-MO ADV-MO  mehr
        ADJD-HD  rechtzeitig
```

[6] Negation As a final step, we move negative particles. If a clause dominates both a finite and infinitival verb, as well as a negative particle (i.e., a word tagged as `PTKNEG`), then the negative particle is moved to directly follow the finite verb.

As an example, the previous example now has the negative particle *nicht* moved, to give the following clause structure:

```
S PPER-SB  Wir
  VMFIN-HD  konnten
  PTKNEG-NG  nicht
  VVINF-HD  einreichen
  PPER-OA  es
  AP-MO ADV-MO  mehr
        ADJD-HD  rechtzeitig
```

4 Experiments

This section describes experiments with the reordering approach. Our baseline is the phrase-based MT system of (Koehn et al., 2003). We trained this system on the Europarl corpus, which consists of 751,088 sentence pairs with 15,256,792 German words and 16,052,269 English words. Translation performance is measured on a 2000 sentence test set from a different part of the Europarl corpus, with average sentence length of 28 words.

We use BLEU scores (Papineni et al., 2002) to measure translation accuracy. We applied our re-

	Annotator 2		
Annotator 1	R	B	E
R	33	2	5
B	2	13	5
E	9	4	27

Table 2: Table showing the level of agreement between two annotators on 100 translation judgements. **R** gives counts corresponding to translations where an annotator preferred the reordered system; **B** signifies that the annotator preferred the baseline system; **E** means an annotator judged the two systems to give equal quality translations.

ordering method to both the training and test data, and retrained the system on the reordered training data. The BLEU score for the new system was 26.8%, an improvement from 25.2% BLEU for the baseline system.

4.1 Human Translation Judgements

We also used human judgements of translation quality to evaluate the effectiveness of the reordering rules. We randomly selected 100 sentences from the test corpus where the English reference translation was between 10 and 20 words in length.[1] For each of these 100 translations, we presented the two annotators with three translations: the reference (human) translation, the output from the baseline system, and the output from the system with reordering. No indication was given as to which system was the baseline system, and the ordering in which the baseline and reordered translations were presented was chosen at random on each example, to prevent ordering effects in the annotators' judgements. For each example, we asked each of the annotators to make one of two choices: 1) an indication that one translation was an improvement over the other; or 2) an indication that the translations were of equal quality.

Annotator 1 judged 40 translations to be improved by the reordered model; 40 translations to be of equal quality; and 20 translations to be worse under the reordered model. Annotator 2 judged 44 translations to be improved by the reordered model; 37 translations to be of equal quality; and 19 translations to be worse under the reordered model. Table 2 gives figures indicating agreement rates between the annotators. Note that if we only consider preferences where both annotators were in agreement (and consider all disagreements to fall into the "equal" category), then 33 translations improved under the reordering system, and 13 translations became worse. Figure 3 shows a random selection of the translations where annotator 1 judged the reordered model to give an improvement; Figure 4 shows examples where the baseline system was preferred by annotator 1. We include these examples to give a qualitative impression of the differences between the baseline and reordered system. Our (no doubt subjective) impression is that the cases in figure 3 are more clear cut instances of translation improvements, but we leave the reader to make his/her own judgement on this point.

[1]We chose these shorter sentences for human evaluation because in general they include a single clause, which makes human judgements relatively straightforward.

4.2 Statistical Significance

We now describe statistical significance tests for our results. We believe that applying significance tests to *Bleu* scores is a subtle issue, for this reason we go into some detail in this section.

We used the sign test (e.g., see page 166 of (Lehmann, 1986)) to test the statistical significance of our results. For a source sentence X, the sign test requires a function $f(X)$ that is defined as follows:

$$f(X) = \begin{cases} + & \text{If reordered system produces a better translation for } X \text{ than the baseline} \\ - & \text{If baseline produces a better translation for } X \text{ than the reordered system.} \\ = & \text{If the two systems produce equal quality translations on } X \end{cases}$$

We assume that sentences X are drawn from some underlying distribution $P(X)$, and that the test set consists of independently, identically distributed (IID) sentences from this distribution. We can define the following probabilities:

$$p_+ = \text{Probability}(f(X) = +) \quad (1)$$
$$p_- = \text{Probability}(f(X) = -) \quad (2)$$

where the probability is taken with respect to the distribution $P(X)$. The sign test has the null hypothesis $H_0 = \{p_+ \leq p_-\}$ and the alternative hypothesis $H_1 = \{p_+ > p_-\}$. Given a sample of n test points $\{X_1, \ldots, X_n\}$, the sign test depends on calculation of the following counts: $c_+ = |\{i : f(X_i) = +\}|$, $c_- = |\{i : f(X_i) = -\}|$, and $c_0 = |\{i : f(X_i) = 0\}|$, where $|\mathcal{S}|$ is the cardinality of the set $\mathcal{S}$.

We now come to the definition of $f(X)$ — how should we judge whether a translation from one system is better or worse than the translation from another system? A critical problem with *Bleu* scores is that they are a function of *an entire test corpus* and do not give translation scores for single sentences. Ideally we would have some measure $f_R(X) \in \mathbb{R}$ of the quality of the translation of sentence X under the reordered system, and a corresponding function $f_B(X)$ that measures the quality of the baseline translation. We could then define $f(X)$ as follows:

$$\begin{array}{ll} f(X) = + & \text{If } f_R(X) > f_B(X) \\ f(X) = - & \text{If } f_R(X) < f_B(X) \\ f(X) = 0 & \text{If } f_R(X) = f_B(X) \end{array}$$

Unfortunately *Bleu* scores do not give per-sentence measures $f_R(X)$ and $f_B(X)$, and thus do not allow a definition of $f(X)$ in this way. In general the lack of per-sentence scores makes it challenging to apply significance tests to *Bleu* scores.[2]

To get around this problem, we make the following approximation. For any test sentence X_i, we calculate $f(X_i)$ as follows. First, we define s to be the *Bleu* score for the test corpus when translated by the baseline model. Next, we define s_i to be the *Bleu* score when all sentences other than X_i are translated by the baseline model, and where X_i itself is translated by the *reordered* model. We then define

$$\begin{array}{ll} f(X_i) = + & \text{If } s_i > s \\ f(X_i) = - & \text{If } s_i < s \\ f(X_i) = 0 & \text{If } s_i = s \end{array}$$

Note that strictly speaking, this definition of $f(X_i)$ is not valid, as it depends on the entire set of sample points $X_1 \ldots X_n$ rather than X_i alone. However, we believe it is a reasonable approximation to an ideal function $f(X)$ that indicates whether the translations have improved or not under the reordered system. Given this definition of $f(X)$, we found that $c_+ = 1057$, $c_- = 728$, and $c_0 = 215$. (Thus 52.85% of all test sentences had improved translations under the baseline system, 36.4% of all sentences had worse translations, and 10.75% of all sentences had the same quality as before.) If our definition of $f(X)$ was correct, these values for c_+ and c_- would be significant at the level $p \leq 0.01$.

We can also calculate confidence intervals for the results. Define P to be the probability that the reordered system improves on the baseline system, given that the two systems do not have equal performance. The relative frequency estimate of P is $\hat{P} = 1057/(1057 + 728) = 59.2\%$. Using a normal approximation (e.g., see Example 6.17 from (Wasserman, 2004)) a 95% confidence interval for a sample size of 1785 is $\hat{P} \pm 2.3\%$, giving a 95% confidence interval of $[56.9\%, 61.5\%]$ for P.

5 Conclusions

We have demonstrated that adding knowledge about syntactic structure can significantly improve the performance of an existing state-of-the-art statistical machine translation system. Our approach makes use of syntactic knowledge to overcome a weakness of tradition SMT systems, namely long-distance reordering. We pose clause restructuring as a problem for machine translation. Our current approach is based on hand-crafted rules, which are based on our linguistic knowledge of how German and English syntax differs. In the future we may investigate data-driven approaches, in an effort to learn reordering models automatically. While our experiments are on German, other languages have word orders that are very different from English, so we believe our methods will be generally applicable.

Acknowledgements

We would like to thank Amit Dubey for providing the German parser used in our experiments. Thanks to Brooke Cowan and Luke Zettlemoyer for providing the human judgements of translation performance. Thanks also to Regina Barzilay for many helpful comments on an earlier draft of this paper. Any remaining errors are of course our own. Philipp Koehn was supported by a grant from NTT, Agmt. dtd. 6/21/1998. Michael Collins was supported by NSF grants IIS-0347631 and IIS-0415030.

[2]The lack of per-sentence scores means that it is not possible to apply standard statistical tests such as the sign test or the t-test (which would test the hypothesis $E[f_R(X)] > E[f_B(X)]$, where $E[.]$ is the expected value under X). Note that previous work (Koehn, 2004; Zhang and Vogel, 2004) has suggested the use of bootstrap tests (Efron and Tibshirani, 1993) for the calculation of confidence intervals for Bleu scores. (Koehn, 2004) gives empirical evidence that these give accurate estimates for Bleu statistics. However, correctness of the bootstrap method relies on some technical properties of the statistic (e.g., Bleu scores) being used (e.g., see (Wasserman, 2004) theorem 8.3); (Koehn, 2004; Zhang and Vogel, 2004) do not discuss whether Bleu scores meet any such criteria, which makes us uncertain of their correctness when applied to Bleu scores.

R:	the current difficulties should encourage us to redouble our efforts to promote cooperation in the euro-mediterranean framework.
C:	the current problems should spur us to intensify our efforts to promote cooperation within the framework of the europa-mittelmeerprozesses.
B:	the current problems should spur us, our efforts to promote cooperation within the framework of the europa-mittelmeerprozesses to be intensified.
R:	propaganda of any sort will not get us anywhere.
C:	with any propaganda to lead to nothing.
B:	with any of the propaganda is nothing to do here.
R:	yet we would point out again that it is absolutely vital to guarantee independent financial control.
C:	however, we would like once again refer to the absolute need for the independence of the financial control.
B:	however, we would like to once again to the absolute need for the independence of the financial control out.
R:	i cannot go along with the aims mr brok hopes to achieve via his report.
C:	i cannot agree with the intentions of mr brok in his report persecuted.
B:	i can intentions, mr brok in his report is not agree with.
R:	on method, i think the nice perspectives, from that point of view, are very interesting.
C:	what the method is concerned, i believe that the prospects of nice are on this point very interesting.
B:	what the method, i believe that the prospects of nice in this very interesting point.
R:	secondly, without these guarantees, the fall in consumption will impact negatively upon the entire industry.
C:	and, secondly, the collapse of consumption without these guarantees will have a negative impact on the whole sector.
B:	and secondly, the collapse of the consumption of these guarantees without a negative impact on the whole sector.
R:	awarding a diploma in this way does not contravene uk legislation and can thus be deemed legal.
C:	since the award of a diploms is not in this form contrary to the legislation of the united kingdom, it can be recognised as legitimate.
B:	since the award of a diploms in this form not contrary to the legislation of the united kingdom is, it can be recognised as legitimate.
R:	i should like to comment briefly on the directive concerning undesirable substances in products and animal nutrition.
C:	i would now like to comment briefly on the directive on undesirable substances and products of animal feed.
B:	i would now like to briefly to the directive on undesirable substances and products in the nutrition of them.
R:	it was then clearly shown that we can in fact tackle enlargement successfully within the eu 's budget.
C:	at that time was clear that we can cope with enlargement, in fact, within the framework drawn by the eu budget.
B:	at that time was clear that we actually enlargement within the framework able to cope with the eu budget, the drawn.

Figure 3: Examples where annotator 1 judged the reordered system to give an improved translation when compared to the baseline system. Recall that annotator 1 judged 40 out of 100 translations to fall into this category. These examples were chosen at random from these 40 examples, and are presented in random order. **R** is the human (reference) translation; **C** is the translation from the system with reordering; **B** is the output from the baseline system.

References

Alshawi, H. (1996). Head automata and bilingual tiling: Translation with minimal representations (invited talk). In *Proceedings of ACL 1996*.

Berger, A. L., Pietra, S. A. D., and Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–69.

Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–313.

Charniak, E., Knight, K., and Yamada, K. (2003). Syntax-based language models for statistical machine translation. In *Proceedings of the MT Summit IX*.

Dubey, A. and Keller, F. (2003). Parsing german with sister-head dependencies. In *Proceedings of ACL 2003*.

Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Springer-Verlag.

Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *Proceedings of HLT-NAACL 2004*.

Gildea, D. (2003). Loosely tree-based alignment for machine translation. In *Proceedings of ACL 2003*.

Graehl, J. and Knight, K. (2004). Training tree transducers. In *Proceedings of HLT-NAACL 2004*.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In Lin, D. and Wu, D., editors, *Proceedings of EMNLP 2004*.

Koehn, P. and Knight, K. (2003). Feature-rich statistical translation of noun phrases. In Hinrichs, E. and Roth, D., editors, *Proceedings of ACL 2003*, pages 311–318.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase based translation. In *Proceedings of HLT-NAACL 2003*.

Lehmann, E. L. (1986). *Testing Statistical Hypotheses (Second Edition)*. Springer-Verlag.

R:	on the other hand non-british hauliers pay nothing when travelling in britain.
C:	on the other hand, foreign kraftverkehrsunternehmen figures anything if their lorries travelling through the united kingdom.
B:	on the other hand, figures foreign kraftverkehrsunternehmen nothing if their lorries travel by the united kingdom.
R:	i think some of the observations made by the consumer organisations are included in the commission 's proposal.
C:	i think some of these considerations, the social organisations will be addressed in the commission proposal.
B:	i think some of these considerations, the social organisations will be taken up in the commission 's proposal.
R:	during the nineties the commission produced several recommendations on the issue but no practical solutions were found.
C:	in the nineties, there were a number of recommendations to the commission on this subject to achieve without, however, concrete results.
B:	in the 1990s, there were a number of recommendations to the commission on this subject without, however, to achieve concrete results.
R:	now, in a panic, you resign yourselves to action.
C:	in the current paniksituation they must react necessity.
B:	in the current paniksituation they must of necessity react.
R:	the human aspect of the whole issue is extremely important.
C:	the whole problem is also a not inconsiderable human side.
B:	the whole problem also has a not inconsiderable human side.
R:	in this area we can indeed talk of a european public prosecutor.
C:	and we are talking here, in fact, a european public prosecutor.
B:	and here we can, in fact speak of a european public prosecutor.
R:	we have to make decisions in nice to avoid endangering enlargement, which is our main priority.
C:	we must take decisions in nice, enlargement to jeopardise our main priority.
B:	we must take decisions in nice, about enlargement be our priority, not to jeopardise.
R:	we will therefore vote for the amendments facilitating its use.
C:	in this sense, we will vote in favour of the amendments which, in order to increase the use of.
B:	in this sense we vote in favour of the amendments which seek to increase the use of.
R:	the fvo mission report mentioned refers specifically to transporters whose journeys originated in ireland.
C:	the quoted report of the food and veterinary office is here in particular to hauliers, whose rushed into shipments of ireland.
B:	the quoted report of the food and veterinary office relates in particular, to hauliers, the transport of rushed from ireland.

Figure 4: Examples where annotator 1 judged the reordered system to give a worse translation than the baseline system. Recall that annotator 1 judged 20 out of 100 translations to fall into this category. These examples were chosen at random from these 20 examples, and are presented in random order. **R** is the human (reference) translation; **C** is the translation from the system with reordering; **B** is the output from the baseline system.

Marcu, D. and Wong, W. (2002). A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP 2002*.

Melamed, I. D. (2004). Statistical machine translation by parsing. In *Proceedings of ACL 2004*.

Niessen, S. and Ney, H. (2004). Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2):181–204.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of ACL 2003*.

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 HLT-NAACL 2004*.

Och, F. J., Tillmann, C., and Ney, H. (1999). Improved alignment models for statistical machine translation. In *Proceedings of EMNLP 1999*, pages 20–28.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*.

Shen, L., Sarkar, A., and Och, F. J. (2004). Discriminative reranking for machine translation. In *Proceedings of HLT-NAACL 2004*.

Wasserman, L. (2004). *All of Statistics*. Springer-Verlag.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).

Xia, F. and McCord, M. (2004). Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of Coling 2004*.

Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of ACL 2001*.

Zhang, Y. and Vogel, S. (2004). Measuring confidence intervals for the machine translation evaluation metrics. In *Proceedings of the Tenth Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*.

Machine Translation Using Probabilistic Synchronous Dependency Insertion Grammars

Yuan Ding **Martha Palmer**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
{yding, mpalmer}@linc.cis.upenn.edu

Abstract

Syntax-based statistical machine translation (MT) aims at applying statistical models to structured data. In this paper, we present a syntax-based statistical machine translation system based on a probabilistic synchronous dependency insertion grammar. Synchronous dependency insertion grammars are a version of synchronous grammars defined on dependency trees. We first introduce our approach to inducing such a grammar from parallel corpora. Second, we describe the graphical model for the machine translation task, which can also be viewed as a stochastic tree-to-tree transducer. We introduce a polynomial time decoding algorithm for the model. We evaluate the outputs of our MT system using the NIST and Bleu automatic MT evaluation software. The result shows that our system outperforms the baseline system based on the IBM models in both translation speed and quality.

1 Introduction

Statistical approaches to machine translation, pioneered by (Brown et al., 1993), achieved impressive performance by leveraging large amounts of parallel corpora. Such approaches, which are essentially stochastic string-to-string transducers, do not explicitly model natural language syntax or semantics. In reality, pure statistical systems sometimes suffer from ungrammatical outputs, which are understandable at the phrasal level but sometimes hard to comprehend as a coherent sentence.

In recent years, syntax-based statistical machine translation, which aims at applying statistical models to structural data, has begun to emerge. With the research advances in natural language parsing, especially the broad-coverage parsers trained from treebanks, for example (Collins, 1999), the utilization of structural analysis of different languages has been made possible. Ideally, by combining the natural language syntax and machine learning methods, a broad-coverage and linguistically well-motivated statistical MT system can be constructed.

However, structural divergences between languages (Dorr, 1994)，which are due to either systematic differences between languages or loose translations in real corpora，pose a major challenge to syntax-based statistical MT. As a result, the syntax based MT systems have to transduce between non-isomorphic tree structures.

(Wu, 1997) introduced a polynomial-time solution for the alignment problem based on synchronous binary trees. (Alshawi et al., 2000) represents each production in parallel dependency trees as a finite-state transducer. Both approaches learn the tree representations directly from parallel sentences, and do not make allowances for non-isomorphic structures. (Yamada and Knight, 2001, 2002) modeled translation as a sequence of tree operations transforming a syntactic tree into a string of the target language.

When researchers try to use syntax trees in both languages, the problem of non-isomorphism must be addressed. In theory, stochastic tree transducers and some versions of synchronous grammars provide solutions for the non-isomorphic tree based transduction problem and hence possible solutions for MT. Synchronous Tree Adjoining Grammars, proposed by (Shieber and Schabes, 1990), were introduced primarily for semantics but were later also proposed for translation. Eisner (2003) proposed viewing the MT problem as a probabilistic synchronous tree substitution grammar parsing

Proceedings of the 43rd Annual Meeting of the ACL, pages 541–548,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

problem. Melamed (2003, 2004) formalized the MT problem as synchronous parsing based on multitext grammars. Graehl and Knight (2004) defined training and decoding algorithms for both generalized tree-to-tree and tree-to-string transducers. All these approaches, though different in formalism, model the two languages using tree-based transduction rules or a synchronous grammar, possibly probabilistic, and using multi-lemma elementary structures as atomic units. The machine translation is done either as a stochastic tree-to-tree transduction or a synchronous parsing process.

However, few of the above mentioned formalisms have large scale implementations. And to the best of our knowledge, the advantages of syntax based statistical MT systems over pure statistical MT systems have yet to be empirically verified.

We believe difficulties in inducing a synchronous grammar or a set of tree transduction rules from large scale parallel corpora are caused by:

1. The abilities of synchronous grammars and tree transducers to handle non-isomorphism are limited. At some level, a synchronous derivation process must exist between the source and target language sentences.
2. The training and/or induction of a synchronous grammar or a set of transduction rules are usually computationally expensive if all the possible operations and elementary structures are allowed. The exhaustive search for all the possible sub-sentential structures in a syntax tree of a sentence is NP-complete.
3. The problem is aggravated by the non-perfect training corpora. Loose translations are less of a problem for string based approaches than for approaches that require syntactic analysis.

Hajic et al. (2002) limited non-isomorphism by n-to-m matching of nodes in the two trees. However, even after extending this model by allowing cloning operations on subtrees, Gildea (2003) found that parallel trees over-constrained the alignment problem, and achieved better results with a tree-to-string model than with a tree-to-tree model using two trees. In a different approach, Hwa et al. (2002) aligned the parallel sentences using phrase based statistical MT models and then projected the alignments back to the parse trees.

This motivated us to look for a more efficient and effective way to induce a synchronous grammar from parallel corpora and to build an MT system that performs competitively with the pure statistical MT systems. We chose to build the synchronous grammar on the parallel dependency structures of the sentences. The synchronous grammar is induced by hierarchical tree partitioning operations. The rest of this paper describes the system details as follows: Sections 2 and 3 describe the motivation behind the usage of dependency structures and how a version of synchronous dependency grammar is learned. This grammar is used as the primary translation knowledge source for our system. Section 4 defines the tree-to-tree transducer and the graphical model for the stochastic tree-to-tree transduction process and introduces a polynomial time decoding algorithm for the transducer. We evaluate our system in section 5 with the NIST/Bleu automatic MT evaluation software and the results are discussed in Section 6.

2 The Synchronous Grammar

2.1 Why Dependency Structures?

According to Fox (2002), dependency representations have the best inter-lingual phrasal cohesion properties. The percentage for head crossings is 12.62% and that of modifier crossings is 9.22%. Furthermore, a grammar based on dependency structures has the advantage of being simple in formalism yet having CFG equivalent formal generative capacity (Ding and Palmer, 2004b).

Dependency structures are inherently lexicalized as each node is one word. In comparison, phrasal structures (treebank style trees) have two node types: terminals store the lexical items and non-terminals store word order and phrasal scopes.

2.2 Synchronous Dependency Insertion Grammars

Ding and Palmer (2004b) described one version of synchronous grammar: Synchronous Dependency Insertion Grammars. A Dependency Insertion Grammars (DIG) is a generative grammar formalism that captures word order phenomena within the dependency representation. In the scenario of two languages, the two sentences in the source and target languages can be modeled as being generated from a synchronous derivation process.

A synchronous derivation process for the two syntactic structures of both languages suggests the level of cross-lingual isomorphism between the two trees (e.g. Synchronous Tree Adjoining Grammars (Shieber and Schabes, 1990)).

Apart from other details, a DIG can be viewed as a tree substitution grammar defined on dependency trees (as opposed to phrasal structure trees). The basic units of the grammar are elementary trees (ET), which are sub-sentential dependency structures containing one or more lexical items. The synchronous version, SDIG, assumes that the isomorphism of the two syntactic structures is at the ET level, rather than at the word level, hence allowing non-isomorphic tree to tree mapping.

We illustrate how the SDIG works using the following pseudo-translation example:

- [*Source*] *The girl kissed her kitty cat.*
- [*Target*] *The girl gave a kiss to her cat.*

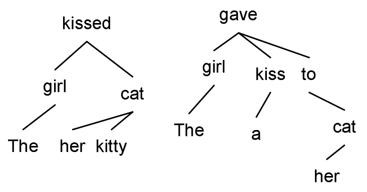


Figure 1. An example

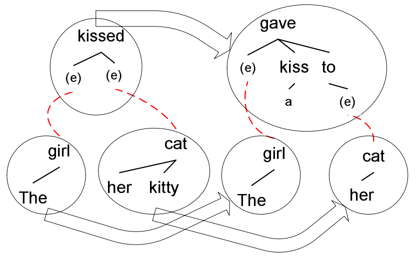


Figure 2. Tree-to-tree transduction

Almost any tree-transduction operations defined on a single node will fail to generate the target sentence from the source sentence without using insertion/deletion operations. However, if we view each dependency tree as an assembly of indivisible sub-sentential elementary trees (ETs), we can find a proper way to transduce the input tree to the output tree. An ET is a single "symbol" in a transducer's language. As shown in Figure 2, each circle stands for an ET and thick arrows denote the transduction of each ET as a single symbol.

3 Inducing a Synchronous Dependency Insertion Grammar

As the start to our syntax-based SMT system, the SDIG must be learned from the parallel corpora.

3.1 Cross-lingual Dependency Inconsistencies

One straightforward way to induce a generative grammar is using EM style estimation on the generative process. Different versions of such training algorithms can be found in (Hajic et al., 2002; Eisner 2003; Gildea 2003; Graehl and Knight 2004).

However, a synchronous derivation process cannot handle two types of cross-language mappings: crossing-dependencies (parent-descendent switch) and broken dependencies (descendent appears elsewhere), which are illustrated below:

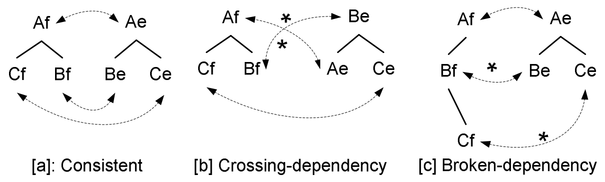


Figure 3. Cross-lingual dependency consistencies

In the above graph, the two sides are English and the foreign dependency trees. Each node in a tree stands for a lemma in a dependency tree. The arrows denote aligned nodes and those resulting inconsistent dependencies are marked with a "*".

Fox (2002) collected the statistics mainly on French and English data: in dependency representations, the percentage of head crossings per chance (case [b] in the graph) is 12.62%.

Using the statistics on cross-lingual dependency consistencies from a small word to word aligned Chinese-English parallel corpus[1], we found that the percentage of crossing-dependencies (case [b]) between Chinese and English is 4.7% while that of broken dependencies (case [c]) is 59.3%.

The large number of broken dependencies presents a major challenge for grammar induction based on a top-down style EM learning process.

Such broken and crossing dependencies can be modeled by SDIG if they appear inside a pair of elementary trees. However, if they appear between the elementary trees, they are not compatible with the isomorphism assumption on which SDIG is based. Nevertheless, the hope is that the fact that the training corpus contains a significant percentage of dependency inconsistencies does not mean that during decoding the target language sentence cannot be written in a dependency consistent way.

3.2 Grammar Induction by Synchronous Hierarchical Tree Partitioning

(Ding and Palmer, 2004a) gave a polynomial time solution for learning parallel sub-sentential de-

[1] Total 826 sentence pairs, 9957 Chinese words, 12660 English words. Data made available by the courtesy of Microsoft Research, Asia and IBM T.J. Watson Research.

pendency structures from non-isomorphic dependency trees. Our approach, while similar to (Ding and Palmer, 2004a) in that we also iteratively partition the parallel dependency trees based on a heuristic function, departs (Ding and Palmer, 2004a) in three ways: (1) we base the hierarchical tree partitioning operations on the categories of the dependency trees; (2) the statistics of the resultant tree pairs from the partitioning operation are collected at each iteration rather than at the end of the algorithm; (3) we do not re-train the word to word probabilities at each iteration. Our grammar induction algorithm is sketched below:

```
Step 0. View each tree as a "bag of words" and train a
        statistical translation model on all the tree pairs to
        acquire word-to-word translation probabilities. In
        our implementation, the IBM Model 1 (Brown et
        al., 1993) is used.
Step 1. Let $i$ denote the current iteration and let
        $C = CategorySequence[i]$ be the current syntactic category set.
For each tree pair in the corpus, do {
  a) For the tentative synchronous partitioning operation, use a heuristic function to select the BEST word
     pair $(e_{i*}, f_{j*})$, where both $e_{i*}$, $f_{j*}$ are NOT "chosen",
     $Category(e_{i*}) \in C$ and $Category(f_{j*}) \in C$.
  b) If $(e_{i*}, f_{j*})$ is found in (a), mark $e_{i*}$, $f_{j*}$ as "chosen" and go back to (a), else go to (c).
  c) Execute the synchronous tree partitioning operation on all the "chosen" word pairs on the tree pair.
     Hence, several new tree pairs are created. Replace the old tree pair with the new tree pairs together with the
     rest of the old tree pair.
  d) Collect the statistics for all the new tree pairs as elementary tree pairs. }
Step 2. $i = i + 1$. Go to Step 1 for the next iteration.
```

At each iteration, one specific set of categories of nodes is handled. The category sequence we used in the grammar induction is:

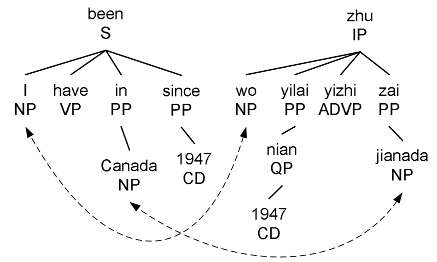
1. *Top-NP*: the noun phrases that do not have another noun phrase as parent or ancestor.
2. *NP*: all the noun phrases
3. *VP, IP, S, SBAR*: verb phrases equivalents.
4. *PP, ADJP, ADVP, JJ, RB*: all the modifiers
5. *CD*: all the numbers.

We first process top *NP* chunks because they are the most stable between languages. Interestingly, NPs are also used as anchor points to learn monolingual paraphrases (Ibrahim et al., 2003). The phrasal structure categories can be extracted from automatic parsers using methods in (Xia, 2001).

An illustration is given below (Chinese in *pinyin* form). The placement of the dependency arcs reflects the relative word order between a parent node and all its immediate children. The collected ETs are put into square boxes and the partitioning operations taken are marked with dotted arrows.

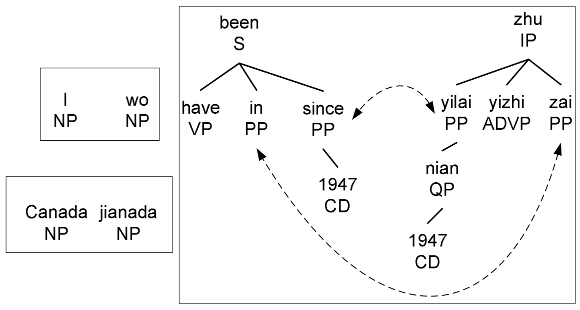
- [*English*] *I have been in Canada since 1947.*
- [*Chinese*] *Wo 1947 nian yilai yizhi zhu zai jianada.*
- [*Glossary*] *I 1947 year since always live in Canada*

[ITERATION 1 & 2] Partition at word pair ("*I*" and "*wo*") ("Canada" and "janada")

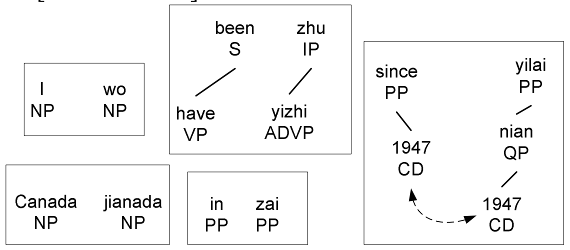


[ITERATION 3] ("*been*" and "*zhu*") are chosen but no partition operation is taken because they are roots.

[ITERATION 4] Partition at word pair ("*since*" and "*yilai*") ("*in*" and "*zai*")



[ITERATION 5] Partition at "*1947*" and "*1947*"



[FINALLY] Total of 6 resultant ET pairs (figure omitted)

Figure 4. An Example

3.3 Heuristics

Similar to (Ding and Palmer, 2004a), we also use a heuristic function in Step 1(a) of the algorithm to rank all the word pairs for the tentative tree partitioning operation. The heuristic function is based on a set of heuristics, most of which are similar to those in (Ding and Palmer, 2004a).

For a word pair (e_i, f_j) for the tentative partitioning operation, we briefly describe the heuristics:

- Inside-outside probabilities: We borrow the idea from PCFG parsing. This is the probability of an English subtree (inside) generating a foreign subtree and the probability of the English residual tree (outside) generating a foreign residual tree. Here both probabilities are based on a "bag of words" model.
- Inside-outside penalties: here the probabilities of the inside English subtree generating the outside foreign residual tree and outside English residual tree generating the inside English subtree are used as penalty terms.
- Entropy: the entropy of the word to word translation probability of the English word e_i.
- Part-of-Speech mapping template: whether the POS tags of the two words are in the "highly likely to match" POS tag pairs.
- Word translation probability: $\mathrm{P}(f_j \mid e_i)$.
- Rank: the rank of the word to word probability of f_j in as a translation of e_i among all the foreign words in the current tree.

The above heuristics are a set of real valued numbers. We use a Maximum Entropy model to interpolate the heuristics in a log-linear fashion, which is different from the error minimization training in (Ding and Palmer, 2004a).

$$\begin{aligned}&\mathrm{P}\left(y \mid h_0(e_i,f_j), h_1(e_i,f_j) \ldots h_n(e_i,f_j)\right)\\&=\frac{1}{Z}\exp\left(\sum_k \lambda_k h_k(e_i,f_j)+\lambda_s\right)\end{aligned} \quad (1)$$

where $y=(0,1)$ as labeled in the training data whether the two words are mapped with each other.

The MaxEnt model is trained using the same word level aligned parallel corpus as the one in Section 3.1. Although the training corpus isn't large, the fact that we only have a handful of parameters to fit eased the problem.

3.4 A Scaled-down SDIG

It is worth noting that the set of derived parallel dependency Elementary Trees is not a full-fledged SDIG yet. Many features in the SDIG formalism such as arguments, head percolation, etc. are not yet filled. We nevertheless use this derived grammar as a Mini-SDIG, assuming the unfilled features as empty by default. A full-fledged SDIG remains a goal for future research.

4 The Machine Translation System

4.1 System Architecture

As discussed before (see Figure 1 and 2), the architecture of our syntax based statistical MT system is illustrated in Figure 5. Note that this is a nondeterministic process. The input sentence is first parsed using an automatic parser and a dependency tree is derived. The rest of the pipeline can be viewed as a stochastic tree transducer. The MT decoding starts first by decomposing the input dependency tree in to elementary trees. Several different results of the decomposition are possible. Each decomposition is indeed a derivation process on the foreign side of SDIG. Then the elementary trees go through a transfer phase and target ETs are combined together into the output.

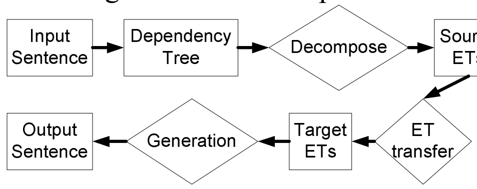


Figure 5. System architecture

4.2 The Graphical Model

The stochastic tree-to-tree transducer we propose models MT as a probabilistic optimization process.

Let f be the input sentence (foreign language), and e be the output sentence (English). We have $\mathrm{P}(e \mid f)=\dfrac{\mathrm{P}(f \mid e)\mathrm{P}(e)}{\mathrm{P}(f)}$, and the best translation is:

$$e^* = \arg\max_e \mathrm{P}(f \mid e)\mathrm{P}(e) \quad (2)$$

$\mathrm{P}(f \mid e)$ and $\mathrm{P}(e)$ are also known as the "translation model" (TM) and the "language model" (LM). Assuming the decomposition of the foreign tree is given, our approach, which is based on ETs, uses the graphical model shown in Figure 6.

In the model, the left side is the input dependency tree (foreign language) and the right side is the output dependency tree (English). Each circle stands for an ET. The solid lines denote the syntactical dependencies while the dashed arrows denote the statistical dependencies.

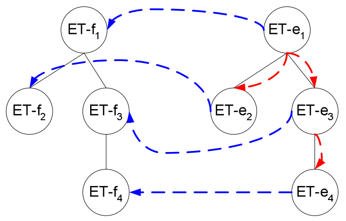


Figure 6
The graphical model

Let $\mathrm{T}(x)$ be the dependency tree constructed from sentence x. A tree-decomposition function $\mathrm{D}(t)$ is defined on a dependency tree t, and outputs a certain ET derivation tree of t, which is generated by decomposing t into ETs. Given t, there could be multiple decompositions. Conditioned on decomposition D, we can rewrite (2) as:

$$\begin{aligned} e^* &= \arg\max_e \sum_D \mathrm{P}(f,e \mid D)\mathrm{P}(D) \\ &= \arg\max_e \sum_D \mathrm{P}(f \mid e,D)\mathrm{P}(e \mid D)\mathrm{P}(D) \end{aligned} \tag{3}$$

By definition, the ET derivation trees of the input and output trees should be isomorphic: $\mathrm{D}(\mathrm{T}(f)) \cong \mathrm{D}(\mathrm{T}(e))$. Let $\mathrm{Tran}(u)$ be a set of possible translations for the ET u. We have:

$$\begin{aligned} \mathrm{P}(f \mid e,D) &= \mathrm{P}(\mathrm{T}(f) \mid \mathrm{P}(\mathrm{T}(e),D) \\ &= \prod_{u\in \mathrm{D}(\mathrm{T}(f)),\ v\in \mathrm{D}(\mathrm{T}(e)),\ v\in \mathrm{Tran}(u)} \mathrm{P}(u \mid v) \end{aligned} \tag{4}$$

For any ET v in a given ET derivation tree d, let $\mathrm{Root}(d)$ be the root ET of d, and let $\mathrm{Parent}(v)$ denote the parent ET of v. We have:

$$\begin{aligned} \mathrm{P}(e \mid D) &= \mathrm{P}(\mathrm{T}(e) \mid D) \\ &= \mathrm{P}\left(\mathrm{Root}\left(\mathrm{D}(\mathrm{T}(e))\right)\right) \cdot \\ &\quad \cdot \left(\prod_{v\in \mathrm{D}(\mathrm{T}(e)), v\neq \mathrm{Root}(\mathrm{D}(\mathrm{T}(e)))} \mathrm{P}(v \mid \mathrm{Parent}(v)) \right) \end{aligned} \tag{5}$$

where, letting $\mathrm{root}(v)$ denote the root word of v,

$$\mathrm{P}\left(v \mid \mathrm{Parent}(v)\right) = \mathrm{P}\left(\mathrm{root}(v) \mid \mathrm{root}\left(\mathrm{Parent}(v)\right)\right) \tag{6}$$

The prior probability of tree decomposition is defined as: $\mathrm{P}\left(\mathrm{D}(\mathrm{T}(f))\right) = \prod_{u\in \mathrm{D}(\mathrm{T}(f))} \mathrm{P}(u)$ (7)

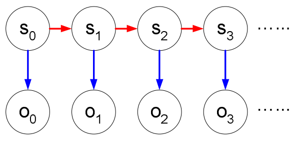


Figure 7
Comparing to the HMM

An analogy between our model and a Hidden Markov Model (Figure 7) may be helpful. In Eq. (4), $\mathrm{P}(u \mid v)$ is analogous to the emission probably $\mathrm{P}(o_i \mid s_i)$ in an HMM. In Eq. (5), $\mathrm{P}(v \mid \mathrm{Parent}(v))$ is analogous to the transition probability $\mathrm{P}(s_i \mid s_{i-1})$ in an HMM. While HMM is defined on a sequence our model is defined on the derivation tree of ETs.

4.3 Other Factors

- Augmenting parallel ET pairs

In reality, the learned parallel ETs are unlikely to cover all the structures that we may encounter in decoding. As a unified approach, we augment the SDIG by adding all the possible word pairs (f_j, e_i) as a parallel ET pair and using the IBM Model 1 (Brown et al., 1993) word to word translation probability as the ET translation probability.

- Smoothing the ET translation probabilities.

The LM probabilities $\mathrm{P}(v \mid \mathrm{Parent}(v))$ are simply estimated using the relative frequencies. In order to handle possible noise from the ET pair learning process, the ET translation probabilities $\mathrm{P}_{emp}(u \mid v)$ estimated by relative frequencies are smoothed using a word level model. For each ET pair (u,v), we interpolate the empirical probability with the "bag of words" probability and then re-normalize:

$$\overline{\mathrm{P}}(u \mid v) = \frac{1}{Z}\mathrm{P}_{emp}(u,v) \cdot \frac{1}{\mathrm{size}(u)^{\mathrm{size}(v)}} \prod_{f_j\in u} \sum_{e_i\in v} \mathrm{P}(f_j \mid e_i) \tag{8}$$

4.4 Polynomial Time Decoding

For efficiency reasons, we use maximum approximation for (3). Instead of summing over all the possible decompositions, we only search for the best decomposition as follows:

$$e^*, D^* = \arg\max_{e,D} \mathrm{P}(f \mid e,D)\mathrm{P}(e \mid D)\mathrm{P}(D) \tag{9}$$

So bringing equations (4) to (9) together, the best translation would maximize:

$$\prod \overline{\mathrm{P}}(u \mid v) \cdot \mathrm{P}\left(\mathrm{Root}(e)\right) \cdot \left(\prod \mathrm{P}(v \mid \mathrm{Parent}(v)) \right) \cdot \prod \mathrm{P}(u) \tag{10}$$

Observing the similarity between our model and a HMM, our dynamic programming decoding algorithm is in spirit similar to the *Viterbi* algorithm except that instead of being sequential the decoding is done on trees in a top down fashion.

As to the relative orders of the ETs, we currently choose not to reorder the children ETs given the parent ET because: (1) the permutation of the ETs is computationally expensive (2) it is possible that we can resort to simple linguistic treatments on the output dependency tree to order the ETs.

Currently, all the ETs are attached to each other at their root nodes.

In our implementation, the different decompositions of the input dependency tree are stored in a shared forest structure, utilizing the dynamic programming property of the tree structures explicitly.

Suppose the input sentence has n words and the shared forest representation has m nodes. Suppose for each word, there are maximally k different ETs containing it, we have $m \leq kn$. Let b be the max breadth factor in the packed forest, it can be shown that the decoder visits at most mb nodes during execution. Hence, we have:

$$T(decoding) \leq O(kbn) \tag{11}$$

which is linear to the input size. Combined with a polynomial time parsing algorithm, the whole decoding process is polynomial time.

5 Evaluation

We implemented the above approach for a Chinese-English machine translation system. We used an automatic syntactic parser (Bikel, 2002) to produce the parallel parse trees. The parser was trained using the Penn English/Chinese Treebanks. We then used the algorithm in (Xia 2001) to convert the phrasal structure trees to dependency trees to acquire the parallel dependency trees. The statistics of the datasets we used are shown as follows:

Dataset	Xinhua	FBIS	NIST
Sentence#	56263	45212	206
Chinese word#	1456495	1185297	27.4 average
English word#	1490498	1611932	37.7 average
Usage	training	training	testing

Figure 8. Evaluation data details

The training set consists of Xinhua newswire data from LDC and the FBIS data (mostly news), both filtered to ensure parallel sentence pair quality. We used the development test data from the 2001 NIST MT evaluation workshop as our test data for the MT system performance. In the testing data, each input Chinese sentence has 4 English translations as references. Our MT system was evaluated using the *n*-gram based Bleu (Papineni et al., 2002) and NIST machine translation evaluation software. We used the NIST software package "mteval" version 11a, configured as case-insensitive.

In comparison, we deployed the GIZA++ MT modeling tool kit, which is an implementation of the IBM Models 1 to 4 (Brown et al., 1993; Al-Onaizan et al., 1999; Och and Ney, 2003). The IBM models were trained on the same training data as our system. We used the ISI Rewrite decoder (Germann et al. 2001) to decode the IBM models.

The results are shown in Figure 9. The score types "I" and "C" stand for individual and cumulative *n*-gram scores. The final NIST and Bleu scores are marked with bold fonts.

Systems	Score Type		1-gram	2-gram	3-gram	4-gram
IBM Model 4	I	NIST	2.562	0.412	0.051	0.008
		Bleu	0.714	0.267	0.099	0.040
	C	NIST	2.562	2.974	3.025	**3.034**
		Bleu	0.470	0.287	0.175	**0.109**
SDIG	I	NIST	5.130	0.763	0.082	0.013
		Bleu	0.688	0.224	0.075	0.029
	C	NIST	5.130	5.892	5.978	**5.987**
		Bleu	0.674	0.384	0.221	**0.132**

Figure 9. Evaluation Results.

The evaluation results show that the NIST score achieved a 97.3% increase, while the Bleu score increased by 21.1%.

In terms of decoding speed, the Rewrite decoder took 8102 seconds to decode the test sentences on a Xeon 1.2GHz 2GB memory machine. On the same machine, the SDIG decoder took 3 seconds to decode, excluding the parsing time. The recent advances in parsing have achieved parsers with $O(n^3)$ time complexity without the grammar constant (McDonald et al., 2005). It can be expected that the total decoding time for SDIG can be as short as 0.1 second per sentence.

Neither of the two systems has any specific translation components, which are usually present in real world systems (E.g. components that translate numbers, dates, names, etc.) It is reasonable to expect that the performance of SDIG can be further improved with such specific optimizations.

6 Discussions

We noticed that the SDIG system outputs tend to be longer than those of the IBM Model 4 system, and are closer to human translations in length.

Translation Type	Human	SDIG	IBM-4
Avg. Sent. Len.	37.7	33.6	24.2

Figure 10. Average Sentence Word Count

This partly explains why the IBM Model 4 system has slightly higher individual *n*-gram precision scores (while the SDIG system outputs are still better in terms of absolute matches).

The relative orders between the parent and child ETs in the output tree is currently kept the same as the orders in the input tree. Admittedly, we benefited from the fact that both Chinese and English are SVO languages, and that many of orderings between the arguments and adjuncts can be kept the same. However, we did notice that this simple "ostrich" treatment caused outputs such as "*foreign financial institutions the president of*".

While statistical modeling of children reordering is one possible remedy for this problem, we believe simple linguistic treatment is another, as the output of the SDIG system is an English dependency tree rather than a string of words.

7 Conclusions and Future Work

In this paper we presented a syntax-based statistical MT system based on a Synchronous Dependency Insertion Grammar and a non-isomorphic stochastic tree-to-tree transducer. A graphical model for the transducer is defined and a polynomial time decoding algorithm is introduced. The results of our current implementation were evaluated using the NIST and Bleu automatic MT evaluation software. The evaluation shows that the SDIG system outperforms an IBM Model 4 based system in both speed and quality.

Future work includes a full-fledged version of SDIG and a more sophisticated MT pipeline with possibly a *tri*-gram language model for decoding.

References

Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, I. D. Melamed, F. Och, D. Purdy, N. A. Smith, and D. Yarowsky. 1999. Statistical machine translation. Technical report, CLSP, Johns Hopkins University.

H. Alshawi, S. Bangalore, S. Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Comp. Linguistics*, 26(1):45-60.

Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *HLT 2002*.

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

Michael John Collins. 1999. Head-driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Ding and Palmer. 2004a. Automatic Learning of Parallel Dependency Treelet Pairs. In First International Joint Conference on NLP (IJCNLP-04).

Ding and Palmer. 2004b. Synchronous Dependency Insertion Grammars: A Grammar Formalism for Syntax Based Statistical MT. Workshop on Recent Advances in Dependency Grammars, COLING-04.

Bonnie J. Dorr. 1994. Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4): 597-633.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In ACL-03. (companion volume), Sapporo, July.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP-02*.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast Decoding and Optimal Decoding for Machine Translation. ACL-01.

Daniel Gildea. 2003. Loosely tree based alignment for machine translation. ACL-03, Japan.

Jonathan Graehl and Kevin Knight. 2004. Training Tree Transducers. In NAACL/HLT-2004

Jan Hajic, et al. 2002. Natural language generation in the context of machine translation. Summer workshop final report, Center for Language and Speech Processing, Johns Hopkins University, Baltimore.

Rebecca Hwa, Philip S. Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. ACL-02

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting Structural Paraphrases from Aligned Monolingual Corpora. In *Proceedings of the Second International Workshop on Paraphrasing* (IWP 2003)

Dan Melamed. 2004. Statistical Machine Translation by Parsing. In ACL-04, Barcelona, Spain.

Dan Melamed. 2003. Multitext Grammars and Synchronous Parsers, In NAACL/HLT-2003.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. ACL-02, Philadelphia, USA.

Ryan McDonald, Koby Crammer and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. ACL-05.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

S. M. Shieber and Y. Schabes. 1990. *Synchronous Tree-Adjoining Grammars*, Proceedings of the 13th COLING, pp. 253-258, August 1990.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):3-403.

Fei Xia. 2001. Automatic grammar generation from two different perspectives. PhD thesis, U. of Pennsylvania.

Kenji Yamada and Kevin Knight. 2001. A syntax based statistical translation model. ACL-01, France.

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. ACL-02, Philadelphia.

Context-dependent SMT Model using Bilingual Verb-Noun Collocation

Young-Sook Hwang
ATR SLT Research Labs
2-2-2 Hikaridai Seika-cho
Soraku-gun Kyoto, 619-0288, JAPAN
youngsook.hwang@atr.jp

Yutaka Sasaki
ATR SLT Research Labs
2-2-2 Hikaridai Seika-cho
Soraku-gun Kyoto, 619-0288, JAPAN
yutaka.sasaki@atr.jp

Abstract

In this paper, we propose a new context-dependent SMT model that is tightly coupled with a language model. It is designed to decrease the translation ambiguities and efficiently search for an optimal hypothesis by reducing the hypothesis search space. It works through reciprocal incorporation between source and target context: a source word is determined by the context of previous and corresponding target words and the next target word is predicted by the pair consisting of the previous target word and its corresponding source word. In order to alleviate the data sparseness in chunk-based translation, we take a stepwise back-off translation strategy. Moreover, in order to obtain more semantically plausible translation results, we use bilingual verb-noun collocations; these are automatically extracted by using chunk alignment and a monolingual dependency parser. As a case study, we experimented on the language pair of Japanese and Korean. As a result, we could not only reduce the search space but also improve the performance.

1 Introduction

For decades, many research efforts have contributed to the advance of statistical machine translation. Recently, various works have improved the quality of statistical machine translation systems by using phrase translation (Koehn et al., 2003; Marcu et al., 2002; Och et al., 1999; Och and Ney, 2000; Zens et al., 2004). Most of the phrase-based translation models have adopted the noisy-channel based IBM style models (Brown et al., 1993):

$$\hat{e}_1^I = argmax_{e_1^I} Pr(f_1^J|e_1^I)Pr(e_1^I) \qquad (1)$$

In these model, we have two types of knowledge: translation model, $Pr(f_1^J|e_1^I)$ and language model, $Pr(e_1^I)$. The translation model links the source language sentence to the target language sentence. The language model describes the well-formedness of the target language sentence and might play a role in restricting hypothesis expansion during decoding. To recover the word order difference between two languages, it also allows modeling the reordering by introducing a relative distortion probability distribution. However, in spite of using such a language model and a distortion model, the translation outputs may not be fluent or in fact may produce nonsense.

To make things worse, the huge hypothesis search space is much too large for an exhaustive search. If arbitrary reorderings are allowed, the search problem is NP-complete (Knight, 1999). According to a previous analysis (Koehn et al., 2004) of how many hypotheses are generated during an exhaustive search using the IBM models, the upper bound for the number of states is estimated by $N \simeq 2^J|V_e|^2J$, where J is the number of source words and $|V_e|$ is the size of the target vocabulary. Even though the number of possible translations of the last two words is much smaller than $|V_e|^2$, we still need to make further improvement. The main concern is the exponential explosion from the possible configurations of source words covered by a hypothesis. In order to reduce the number of possible configurations of source words, decoding algorithms based on A^* as well as the beam search algorithm have been proposed (Koehn et al., 2004; Och et al., 2001). (Koehn et al., 2004; Och et al., 2001) used heuristics for pruning implausible hypotheses.

Our approach to this problem examines the possibility of utilizing context information in a given language pair. Under a given target context, the corresponding source word of a given target word is almost deterministic. Conversely, if a translation pair is given, then the related target or source context is predictable. This implies that if we considered bilingual context information in a given language pair during decoding, we can reduce the computational complexity of the hypothesis search; specifically, we could reduce the possible configurations of source words as well as the number of possible target translations.

In this study, we present a statistical machine translation model as an alternative to the classical IBM-style model. This model is tightly coupled with target language model and utilizes bilingual context information. It is designed to not only reduce the hypothesis search space by decreasing the translation ambiguities but also improve translation performance. It works through reciprocal incorporation between source and target context: source words are determined by the context of previous and corresponding target words, and the next target words are predicted by the current translation pair. Accordingly, we do not need to consider any distortion model or language model as is the case with IBM-style models.

Under this framework, we propose a chunk-based translation model for more grammatical, fluent and accurate output. In order to alleviate the data sparseness problem in chunk-based translation, we use a stepwise back-off method in the order of a chunk, sub-parts of the chunk, and word level. Moreover, we utilize verb-noun collocations in dealing with long-distance dependency which are automatically extracted by using chunk alignment and a monolingual dependency parser.

As a case study, we developed a Japanese-to-Korean translation model and performed some experiments on the BTEC corpus.

2 Overview of Translation Model

The goal of machine translation is to transfer the meaning of a source language sentence, $f_1^J = f_1 \dots f_J$, into a target language sentence, $e_1^I = e_1 \dots e_I$. In most types of statistical machine translation, conditional probability $Pr(e_1^I|f_1^J)$ is used to describe the correspondence between two sentences. This model is used directly for translation by solving the following maximization problem:

$$\begin{aligned} \hat{e}_1^I &= argmax_{e_1^I} Pr(e_1^I|f_1^J) & (2)\\ &= argmax_{e_1^I} \frac{Pr(e_1^I, f_1^J)}{Pr(f_1^J)} & (3)\\ &= argmax_{e_1^I} Pr(e_1^I, f_1^J) & (4)\end{aligned}$$

Since a source language sentence is given and the $Pr(f_1^J)$ probability is applied to all possible corresponding target sentences, we can ignore the denominator in equation (3). As a result, the joint probability model can be used to describe the correspondence between two sentences. We apply Markov chain rules to the joint probability model and obtain the following decomposed model:

$$Pr(e_1^I, f_1^J) \simeq \prod_{i=1}^{I} Pr(f_{a_i}|e_i, e_{i-1}) Pr(e_i|e_{i-1} f_{a_{i-1}}) \quad (5)$$

where a_i is the index of the source word that is aligned to the word e_i under the assumption of the fixed one-to-one alignment. In this model, we have two probabilities:

- source word prediction probability under a given target language context, $Pr(f_{a_i}|e_{i-1}, e_i)$
- target word prediction probability under the preceding translation pair, $Pr(e_i|e_{i-1}, f_{a_{i-1}})$

The probability of target word prediction is used for selecting the target word that follows the previous target words. In order to make this more deterministic, we use bilingual context, i.e. the translation pair of the preceding target word. For a given target word, the corresponding source word is predicted by source word prediction probability based on the current and preceding target words.

Since a target and a source word are predicted through reciprocal incorporation between source and target context from the beginning of a target sentence, the word order in the target sentence is automatically determined and the number of possible configurations of source words is decreased. Thus, we do not need to perform any computation for word re-ordering. Moreover, since correspondences are provided based on bilingual contextual evidence, translation ambiguities can be decreased. As a result, the proposed model is expected to reduce computational complexity during the decoding as well as improve performance.

Furthermore, since a word-based translation approach is often incapable of handling complicated expressions such as an idiomatic expressions or complicated verb phrases, it often outputs nonsense translations. To avoid nonsense translations and to increase explanatory power, we incorporate structural aspects of the language into the chunk-based translation model. In our model, one source chunk is translated by exactly one target chunk, i.e., one-to-one chunk alignment. Thus we obtain:

$$\tilde{e}_1^K = argmax_{\tilde{e}_1^K} Pr(\tilde{e}_1^K, \tilde{f}_1^K) \tag{6}$$

$$Pr(\tilde{e}_1^K, \tilde{f}_1^K) \simeq \prod_{i=1}^{K} Pr(\tilde{f}_{a_i}|\tilde{e}_i, \tilde{e}_{i-1}) Pr(\tilde{e}_i|\tilde{e}_{i-1}, \tilde{f}_{a_{i-1}}) \tag{7}$$

where K is the number of chunks in a source and a target sentence.

3 Chunk-based J/K Translation Model with Back-Off

With the translation framework described above, we built a chunk-based J/K translation model as a case study. Since a chunk-based translation model causes severe data sparseness, it is often impossible to obtain any translation of a given source chunk. In order to alleviate this problem, we apply back-off translation models while giving the consideration to linguistic characteristics.

Japanese and Korean is a very close language pair. Both are agglutinative and inflected languages in the word formation of a *bunsetsu* and an *eojeol*. A *bunsetsu/eojeol* consists of two sub parts: the head part composed of content words and the tail part composed of functional words agglutinated at the end of the head part. The head part is related to the meaning of a given segment, while the tail part indicates a grammatical role of the head in a given sentence.

By putting this linguistic knowledge to practical use, we build a head-tail based translation model as a back-off version of the chunk-based translation model. We place several constraints on this head-tail based translation model as follows:

- The head of a given source chunk corresponds to the head of a target chunk. The tail of the source chunk corresponds to the tail of a target chunk. If a chunk does not have a tail part, we assign *NUL* to the tail of the chunk.

- The head of a given chunk follows the tail of the preceding chunk and the tail follows the head of the given chunk.

The constraints are designed to maintain the structural consistency of a chunk. Under these constraints, the head-tail based translation can be formulated as the following equation:

$$\begin{aligned} Pr(\tilde{f}_{a_i}|\tilde{e}_i, \tilde{e}_{i-1}) Pr(\tilde{e}_i|\tilde{e}_{i-1}, \tilde{f}_{a_{i-1}}) = \\ Pr(\tilde{f}_{a_i}^h|\tilde{e}_i^h, \tilde{e}_{i-1}^t) Pr(\tilde{e}_i^h|\tilde{e}_{i-1}^t \tilde{f}_{a_{i-1}}^t) \\ Pr(\tilde{f}_{a_i}^t|\tilde{e}_i^t, \tilde{e}_i^h) Pr(\tilde{e}_i^t|\tilde{e}_i^h \tilde{f}_{a_i}^h) \end{aligned} \tag{8}$$

where $\tilde{e}_i^h$ denotes the head of the i^{th} chunk and $\tilde{e}_i^t$ means the tail of the chunk.

In the worst case, even the head-tail based model may fail to obtain translations. In this case, we back it off into a word-based translation model. In the word-based translation model, the constraints on the head-tail based translation model are not applied. The concept of the chunk-based J/K translation framework with back-off scheme can be summarized as follows:

1. Input a dependency-parsed sentence at the chunk level,

2. Apply the chunk-based translation model to the given sentence,

3. If one of chunks does not have any corresponding translation:

    - divide the failed chunk into a head and a tail part,

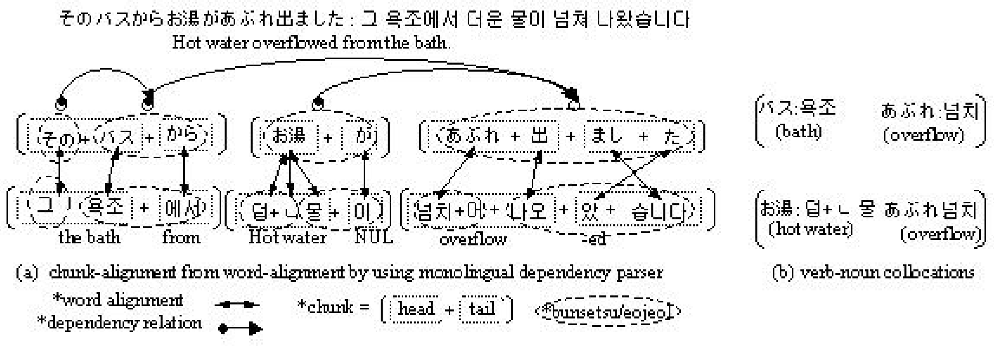

そのバスからお湯があふれ出ました : 그 욕조에서 더운 물이 넘쳐 나왔습니다
Hot water overflowed from the bath.
the bath
from
Hot water
NUL
overflow
-ed
(a) chunk-alignment from word-alignment by using monolingual dependency parser
*word alignment
*dependency relation
*chunk = head + tail
<bunsetsu/eojeol>
バス:욕조 (bath)
あふれ:넘치 (overflow)
お湯:덥+ㄴ 물 あふれ넘치 (hot water) (overflow)
(b) verb-noun collocations

Figure 1: An example of (a) chunk alignment for chunk-based, head-tail based translation and (b) bilingual verb-noun collocation by using the chunk alignment and a monolingual dependency parser

- back-off the translation into the head-tail based translation model,
- if the head or tail does not have any corresponding translation, apply a word-based translation model to the chunk.

Here, the back-off model is applied only to the part that failed to get translation candidates.

3.1 Learning Chunk-based Translation

We learn chunk alignments from a corpus that has been word-aligned by a training toolkit for word-based translation models: the Giza++ (Och and Ney, 2000) toolkit for the IBM models (Brown et al., 1993). For aligning chunk pairs, we consider word(bunsetsu/eojeol) sequences to be chunks if they are in an immediate dependency relationship in a dependency tree. To identify chunks, we use a word-aligned corpus, in which source language sentences are annotated with dependency parse trees by a dependency parser (Kudo et al., 2002) and target language sentences are annotated with POS tags by a part-of-speech tagger (Rim, 2003). If a sequence of target words is aligned with the words in a single source chunk, the target word sequence is regarded as one chunk corresponding to the given source chunk. By applying this method to the corpus, we obtain a word- and chunk-aligned corpus (see Figure 1).

From the aligned corpus, we directly estimate the phrase translation probabilities, $Pr(\tilde{f}|\tilde{e})$, and the model parameters, $Pr(\tilde{f}_{a_i}|\tilde{e}_i, \tilde{e}_{i-1})$, $Pr(\tilde{e}_i|\tilde{e}_{i-1}, \tilde{f}_{a_{i-1}})$. These estimation are made based on relative frequencies.

3.2 Decoding

For efficient decoding, we implement a multi-stack decoder and a beam search with A^* algorithm. At each search level, the beam search moves through at most n-best translation candidates, and a multi-stack is used for partial translations according to the translation cardinality. The output sentence is generated from left to right in the form of partial translations.

Initially, we get n translation candidates for each source chunk with the beam size n. Every possible translation is sorted according to its translation probability. We start the decoding with the initialized beams and initial stack S_0, the top of which has the information of the initial hypothesis, $\langle \tilde{e}_0 = \$, \tilde{f}_0 = \$\rangle$. The decoding algorithm is described in Table 1.

In the decoding algorithm, estimating the backward score is so complicated that the computational complexity becomes too high because of the context consideration. Thus, in order to simplify this problem, we assume the context-independence of only the backward score estimation. The backward score is estimated by the translation probability and language model score of the uncovered segments. For each uncovered segment, we select the best translation with the highest score by multiplying the translation probability of the segment by its language model score. The translation probability and language model score are computed without giving consideration to context.

After estimating the forward and backward score of each partial translation on stack S_i, we try to

```
1. Push the initial hypothesis $\langle \tilde{e}_0 = \$, \tilde{f}_0 = \$ \rangle$ on the initial stack $S_0$
2. for i=1 to K
     • Pop the previous state information of $\langle \tilde{e}_{i-1}, \tilde{f}_{a_{i-1}} \rangle$ from stack $S_{i-1}$
     • Get next target $\tilde{e}_i$ and corresponding source $\tilde{f}_{a_i}$
     • for all pairs of $\langle \tilde{e}_i, \tilde{f}_{a_i} \rangle$
         – Check the head-tail consistency
         – Mark the source segment as a covered one
         – Estimate forward and backward score
         – Push the state of pair $\langle \tilde{e}_i, \tilde{f}_{a_i} \rangle$ onto stack $S_i$
     • Sort all translations on stack $S_i$ by the scores
     • Prune the hypotheses
3. while (stack $S_K$ is not empty)
     • Pop the state of the pair $\langle \tilde{e}_K, \tilde{f}_{a_K} \rangle$
     • Compose translation output, $\langle \tilde{e}_1 \ldots \tilde{e}_K \rangle$
4. Output the best $N$ translations
```

Table 1: A^* multi-stack decoding algorithm

prune the hypotheses. In pruning, we first sort the partial translations on stack S_i according to their scores. If the gradient of scores steeply decreases over the given threshold at the k^{th} translation, we prune the translations of lower scores than the k^{th} one. Moreover, if the number of filtered translations is larger than N, we only take the top N translations. As a final translation, we output the single best translation.

4 Resolving Long-distance Dependency

Since most of the current translation models take only the local context into account, they cannot account for long-distance dependency. This often causes syntactically or semantically incorrect translation to be output. In this section, we describe how this problem can be solved. For handling the long-distance dependency problem, we utilize bilingual verb-noun collocations that are automatically acquired from the chunk-aligned bilingual corpora.

4.1 Automatic Extraction of Bilingual Verb-Noun Collocation(BiVN)

To automatically extract the bilingual verb-noun collocations, we utilize a monolingual dependency parser and the chunk alignment result. The basic concept is the same as that used in (Hwang et al., 2004): bilingual dependency parses are obtained by sharing the dependency relations of a monolingual dependency parser among the aligned chunks. Then bilingual verb sub-categorization patterns are acquired by navigating the bilingual dependency trees. A verb sub-categorization is the collocation of a verb and all of its argument/adjunct nouns, i.e. verb-noun collocation(see Figure 1).

To acquire more reliable and general knowledge, we apply the following filtering method with statistical χ^2 test and unification operation:

- step 1. Filter out the reliable translation correspondences from all of the alignment pairs by χ^2 test at a probability level of α_1

- step 2. Filter out reliable bilingual verb-noun collocations *BiVN* by a unification and χ^2 test at a probability level of α_2: Here, we assume that two bilingual pairs, $\langle v_f : v_e \rangle$ and $\langle n_f : n_e \rangle$ are unifiable into a frame $\langle v_f : v_e, n_f : n_e \rangle$ iff both of them are reliable pairs filtered in step 1. and they share the same verb pair $\langle v_f : v_e \rangle$.

4.2 Application of BiVN

The acquired BiVN is used to evaluate the bilingual correspondence of a verb-noun pair dependent on each other and to select the correct translation. It can be applied to any verb-noun pair regardless of the distance between them in a sentence. Moreover, since the verb-noun relation in BiVN is bilingual knowledge, the sense of each corresponding verb and noun can be almost completely disambiguated by each other.

In our translation system, we apply this **BiVN** during decoding as follows:

1. Pivot verbs and their dependents in a given dependency-parsed source sentence

2. When extending a hypothesis, if one of the pivoted verb and noun pairs is covered and its corresponding translation pair is in **BiVN**, we give positive weight $\beta > 1$ to the hypothesis.

$$\psi(BiVN_i) = \begin{cases} 1 & \text{if } BiVN_i \in \mathbf{BiVN} \\ 0 & \text{otherwise} \end{cases}$$

where $BiVN_i = \langle v_f : v_e, n_f : n_e \rangle$ and $\psi(BiVN_i)$ is a function that indicates whether the bilingual translation pair is in **BiVN**. By adding the weight of the $\psi(BiVN_i)$ function, we refine our model as follows:

$$\tilde{e}_1^K \simeq argmax \quad \prod_{i=1}^{K} Pr(\tilde{f}_{a_i}|\tilde{e}_i, \tilde{e}_{i-1}) \quad Pr(\tilde{e}_i|\tilde{e}_{i-1}\tilde{f}_{a_{i-1}})\beta^{VN(f_{a_i})\psi(BiVN_i)} \tag{10}$$

where $VN(f_{a_i})$ is a function indicating whether the pair of a verb and its argument $\langle v_f, n_f \rangle$ is covered with $v_f = f_{a_i}$ or $n_f = f_{a_i}$ and $BiVN_i = \langle v_f : v_e, n_f : n_e \rangle$ is a bilingual translation pair in the hypothesis.

5 Experiments

5.1 Corpus

The corpus for the experiment was extracted from the Basic Travel Expression Corpus (BTEC), a collection of conversational travel phrases for Japanese and Korean (see Table 2). The entire corpus was split into two parts: 162,320 sentences in parallel for training and 10,150 sentences for test. The Japanese sentences were automatically dependency-parsed by CaboCha (Kudo et al., 2002) and the Korean sentences were automatically POS tagged by KUTagger (Rim, 2003)

5.2 Translation Systems

Four translation systems were implemented for evaluation: 1) Word based IBM-style SMT System(WBIBM), 2) Chunk based IBM-style SMT System(CBIBM), 3) Word based LM tightly Coupled SMT System(WBLMC), and 4) Chunk based LM tightly Coupled SMT System(CBLMC). To examine the effect of BiVN, BiVN was optionally used for each system.

The word-based IBM-style (WBIBM) system[1] consisted of a word translation model and a bi-gram language model. The bi-gram language model was generated by using CMU LM toolkit (Clarkson et al., 1997). Instead of using a fertility model, we allowed a multi-word target of a given source word if it aligned with more than one word. We didn't use any distortion model for word re-ordering. And we used a log-linear model $Pr(e|f) = exp(\sum_i \lambda_i h(e,f))$ for weighting the language model and the translation model. For decoding, we used a multi-stack decoder based on the A^* algorithm, which is almost the same as that described in Section 3. The difference is the use of the language model for controlling the generation of target translations.

[1]In this experiment, a word denotes a morpheme

The chunk-based IBM-style (CBIBM) system consisted of a chunk translation model and a bi-gram language model. To alleviate the data sparseness problem of the chunk translation model, we applied the back-off method at the head-tail or morpheme level. The remaining conditions are the same as those for WBIBM.

The word-based LM tightly coupled (WBLMC) system was implemented for comparison with the chunk-based systems. Except for setting the translation unit as a morpheme, the other conditions are the same as those for the proposed chunk-based translation system.

The chunk-based LM tightly coupled (CBLMC) system is the proposed translation system. A bi-gram language model was used for estimating the backward score.

5.3 Evaluation

Translation evaluations were carried out on 510 sentences selected randomly from the test set. The metrics for the evaluations are as follows:

PER(Position independent WER), which penalizes without considering positional disfluencies(Niesen et al., 2000).

mWER(multi-reference Word Error Rate), which is based on the minimum edit distance between the target sentence and the sentences in the reference set (Niesen et al., 2000).

BLEU, which is the ratio of the n-gram for the translation results found in the reference translations with a penalty for too short sentences (Papineni et al., 2001).

NIST which is a weighted n-gram precision in combination with a penalty for too short sentences.

For this evaluation, we made 10 multiple references available. We computed all of the above criteria with respect to these multiple references.

	Training		Test	
	Japanese	Korean	Japanese	Korean
# of sentences	162,320		10,150	
# of total morphemes	1,153,954	1,179,753	74,366	76,540
# of bunsetsu/eojeol	448,438	587,503	28,882	38,386
vocabulary size	15,682	15,726	5,144	4,594

Table 2: Statistics of Basic Travel Expression Corpus

	PER	mWER	BLEU	NIST
WBIBM	0.3415 / 0.3318	0.3668 / 0.3591	0.5747 / 0.5837	6.9075 / 7.1110
WBLMC	0.2667 / 0.2666	0.2998 / 0.2994	0.5681 / 0.5690	9.0149 / 9.0360
CBIBM	0.2677 / 0.2383	0.2992 / 0.2700	0.6347 / 0.6741	8.0900 / 8.6981
CBLMC	0.1954 / 0.1896	0.2176 / 0.2129	0.7060 / 0.7166	9.9167 / 10.027

Table 3: Evaluation Results of Translation Systems: without BiVN/with BiVN

WBIBM	WBLMC	CBIBM	CBLMC
0.8110 / 0.8330	2.5585 / 2.5547	0.3345 / 0.3399	0.9039 / 0.9052

Table 4: Translation Speed of Each Translation Systems(sec./sentence): without BiVN/with BiVN

5.4 Analysis and Discussion

Table 3 shows the performance evaluation of each system. CBLMC outperformed CBIBM in overall evaluation criteria. WBLMC showed much better performance than WBIBM in most of the evaluation criteria except for *BLEU* score. The interesting point is that the performance of WBLMC is close to that of CBIBM in *PER* and *mWER*. The *BLEU* score of WBLMC is lower than that of CBIBM, but the *NIST* score of WBLMC is much better than that of CBIBM.

The reason the proposed model provided better performance than the IBM-style models is because the use of contextual information in CBLMC and WBLMC enabled the system to reduce the translation ambiguities, which not only reduced the computational complexity during decoding, but also made the translation accurate and deterministic. In addition, chunk-based translation systems outperformed word-based systems. This is also strong evidence of the advantage of contextual information.

To evaluate the effectiveness of bilingual verb-noun collocations, we used the BiVN filtered with $\alpha 1 = .05, \alpha 2 = .1$, where coverage is 64.86% on the test set and average ambiguity is 2.99. We suffered a slight loss in the speed by using the BiVN(see Table 4), but we could improve performance in all of the translation systems(see Table 3). In particular, the performance improvement in CBIBM with BiVN was remarkable. This is a positive sign that the BiVN is useful for handling the problem of long-distance dependency. From this result, we believe that if we increased the coverage of BiVN and its accuracy, we could improve the performance much more.

Table 4 shows the translation speed of each system. For the evaluation of processing time, we used the same machine, with a Xeon 2.8 GHz CPU and 4GB memory , and checked the time of the best performance of each system. The chunk-based translation systems are much faster than the word-based systems. It may be because the translation ambiguities of the chunk-based models are lower than those of the word-based models. However, the processing speed of the IBM-style models is faster than the proposed model. This tendency can be analyzed from two viewpoints: decoding algorithm and DB system for parameter retrieval. Theoretically, the computational complexity of the proposed model is lower than that of the IBM models. The use of a sorting and pruning algorithm for partial translations provides shorter search times in all system. Since the number of parameters for the proposed model is much more than for the IBM-style models, it took a longer time to retrieve parameters. To decrease the processing time, we need to construct a more efficient DB system.

6 Conclusion

In this paper, we proposed a new chunk-based statistical machine translation model that is tightly coupled with a language model. In order to alleviate the data sparseness in chunk-based translation, we applied the back-off translation method at the head-tail and morpheme levels. Moreover, in order to get more semantically plausible translation results by considering long-distance dependency, we utilized verb-noun collocations which were automatically extracted by using chunk alignment and a monolingual dependency parser. As a case study, we experimented on the language pair of Japanese and Korean. Experimental results showed that the proposed translation model is very effective in improving performance. The use of bilingual verb-noun collocations is also useful for improving the performance.

However, we still have some problems of the data sparseness and the low coverage of bilingual verb-noun collocation. In the near future, we will try to solve the data sparseness problem and to increase the coverage and accuracy of verb-noun collocations.

References

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.

P.R. Clarkson and R. Rosenfeld. 1997. *Statistical Language Modeling Using the CMU-Cambridge Toolkit*, Proc. of ESCA Eurospeech.

Young-Sook Hwang, Kyonghee Paik, and Yutaka Sasaki. 2004. *Bilingual Knowledge Extraction Using Chunk Alignment*, Proc. of the 18th Pacific Asia Conference on Language, Information and Computation (PACLIC-18), pp. 127-137, Tokyo.

Kevin Knight. 1999. *Decoding Complexity in Word-Replacement Translation Models, Computational Linguistics, Squibs Discussion*, 25(4).

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003 *Statistical Phrase-Based Translation*, Proc. of the Human Language Technology Conference(HLT/NAACL)

Philipp Koehn. 2004 *Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models*, Proc. of AMTA'04

Taku Kudo, Yuji Matsumoto. 2002. *Japanese Dependency Analyisis using Cascaded Chunking*, Proc. of CoNLL-2002

Daniel Marcu and William Wong. 2002. *A phrase-based, joint probability model for statistical machine translation* , Proc. of EMNLP.

Sonja Niesen, Franz Josef Och, Gregor Leusch, Hermann Ney. 2000. *An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research*, Proc. of the 2nd International Conference on Language Resources and Evaluation, pp. 39-45, Athens, Greece.

Franz Josef Och, Christoph Tillmann, Hermann Ney. 1999. *Improved alignment models for statistical machine translation*, Proc. of EMNLP/WVLC.

Franz Josef Och and Hermann Ney. 2000. *Improved Statistical Alignment Models* , Proc. of the 38th Annual Meeting of the Association for Computational Linguistics, pp. 440-447, Hongkong, China.

Franz Josef Och, Nicola Ueffing, Hermann Ney. 2001. *An Efficient A* Search Algorithm for Statistical Machine Translation* , Data-Driven Machine Translation Workshop, pp. 55-62, Toulouse, France.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. *Bleu: a method for automatic evaluation of machine translation* , IBM Research Report, RC22176.

Toshiyuki Takezawa, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. 2002. *Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world, Proc. of LREC 2002*, pp. 147-152, Spain.

Richard Zens and Hermann Ney. 2004. *Improvements in Phrase-Based Statistical Machine Translation*, Proc. of the Human Language Technology Conference (HLT-NAACL) , Boston, MA, pp. 257-264.

Hae-Chang Rim. 2003. *Korean Morphological Analyzer and Part-of-Speech Tagger*, Technical Report, NLP Lab. Dept. of Computer Science and Engineering, Korea University

A Localized Prediction Model for Statistical Machine Translation

Christoph Tillmann and Tong Zhang
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598 USA
{ctill,tzhang}@us.ibm.com

Abstract

In this paper, we present a novel training method for a localized phrase-based prediction model for statistical machine translation (SMT). The model predicts blocks with orientation to handle local phrase re-ordering. We use a maximum likelihood criterion to train a log-linear block bigram model which uses real-valued features (e.g. a language model score) as well as binary features based on the block identities themselves, e.g. block bigram features. Our training algorithm can easily handle millions of features. The best system obtains a 18.6 % improvement over the baseline on a standard Arabic-English translation task.

1 Introduction

In this paper, we present a block-based model for statistical machine translation. A block is a pair of phrases which are translations of each other. For example, Fig. 1 shows an Arabic-English translation example that uses 4 blocks. During decoding, we view translation as a block segmentation process, where the input sentence is segmented from left to right and the target sentence is generated from bottom to top, one block at a time. A monotone block sequence is generated except for the possibility to swap a pair of neighbor blocks. We use an orientation model similar to the lexicalized block re-ordering model in (Tillmann, 2004; Och et al., 2004): to generate a block b with orientation o relative to its predecessor block b'. During decoding, we compute the probability $P(b_1^n, o_1^n)$ of a block sequence b_1^n with orientation o_1^n as a product of block bigram probabilities:

$$P(b_1^n, o_1^n) \quad \approx \quad \prod_{i=1}^{n} p(b_i, o_i | b_{i-1}, o_{i-1}), \qquad (1)$$

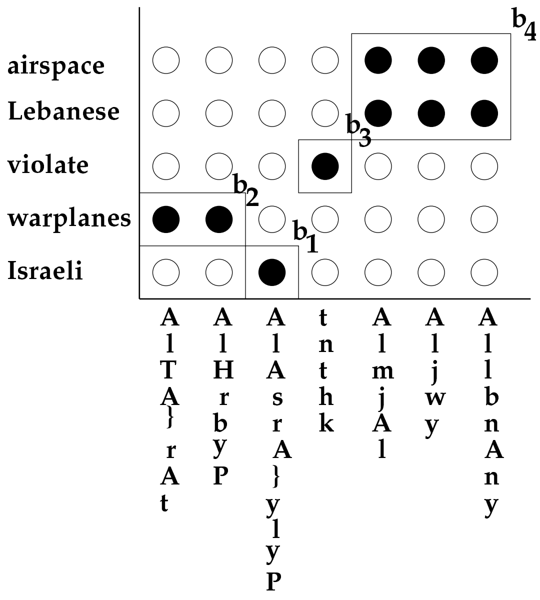


Figure 1: An Arabic-English block translation example, where the Arabic words are romanized. The following orientation sequence is generated: $o_1 = N, o_2 = L, o_3 = N, o_4 = R$.

where b_i is a block and $o_i \in \{L(\text{eft}), R(\text{ight}), N(\text{eutral})\}$ is a three-valued orientation component linked to the block b_i (the orientation o_{i-1} of the predecessor block is currently ignored.). Here, the block sequence with orientation (b_1^n, o_1^n) is generated under the restriction that the concatenated source phrases of the blocks b_i yield the input sentence. In modeling a block sequence, we emphasize adjacent block neighbors that have **Right** or **Left** orientation. Blocks with neutral orientation are supposed to be less strongly 'linked' to their predecessor block and are handled separately. During decoding, most blocks have right orientation $(o = R)$, since the block translations are mostly monotone.

The focus of this paper is to investigate issues in discriminative training of decoder parameters. Instead of directly minimizing error as in earlier work (Och, 2003), we decompose the decoding process into a sequence of local decision steps based on Eq. 1, and then train each local decision rule using convex optimization techniques. The advantage of this approach is that it can easily handle a large amount of features. Moreover, under this view, SMT becomes quite similar to sequential natural language annotation problems such as part-of-speech tagging, phrase chunking, and shallow parsing.
The paper is structured as follows: Section 2 introduces the concept of block orientation bigrams. Section 3 describes details of the localized log-linear prediction model used in this paper. Section 4 describes the online training procedure and compares it to the well known perceptron training algorithm (Collins, 2002). Section 5 shows experimental results on an Arabic-English translation task. Section 6 presents a final discussion.

2 Block Orientation Bigrams

This section describes a phrase-based model for SMT similar to the models presented in (Koehn et al., 2003; Och et al., 1999; Tillmann and Xia, 2003). In our paper, phrase pairs are named blocks and our model is designed to generate block sequences. We also model the position of blocks relative to each other: this is called orientation. To define block sequences with orientation, we define the notion of block orientation bigrams. Starting point for collecting these bigrams is a block set $\Gamma = \{b = (S,T) = (s_1^J, t_1^I)\}$. Here, b is a block consisting of a source phrase S and a target phrase T. J is the source phrase length and I is the target phrase length. Single source and target words are denoted by s_j and t_i respectively, where $j = 1, \cdots, J$ and $i = 1, \cdots, I$. We will also use a special single-word block set $\Gamma_1 \subseteq \Gamma$ which contains only blocks for which $J = I = 1$. For the experiments in this paper, the block set is the one used in (Al-Onaizan et al., 2004). Although this is not investigated in the present paper, different blocksets may be used for computing the block statistics introduced in this paper, which may effect translation results.

For the block set Γ and a training sentence pair, we carry out a two-dimensional pattern matching algorithm to find adjacent matching blocks along with their position in the coordinate system defined by source and target positions (see Fig. 2). Here, we do not insist on a consistent block coverage as one would do during decoding. Among the matching blocks, two blocks b' and b are adjacent if the target phrases T and T' as well as the source phrases S and S' are adjacent. b' is predecessor of block b if b' and b are adjacent and b' occurs below b. A right adjacent successor block b is said to have right orientation $o = R$. A left adjacent successor block is said to have left orientation $o = L$. There are matching blocks b that have no predecessor, such a block has neutral orientation ($o = N$). After matching blocks for a training sentence pair, we look for adjacent block pairs to collect block bigram orientation events e of the type $e = (b', o, b)$. Our model to be presented in Section 3 is used to predict a **future** block orientation pair (b, o) given its predecessor block **history** b'. In Fig. 1, the following block orientation bigrams occur: $(\cdot, N, b_1)$,(b_1, L, b_2),$(\cdot, N, b_3)$,(b_3, R, b_4). Collecting orientation bigrams on all parallel sentence pairs, we obtain an orientation bigram list e_1^N:

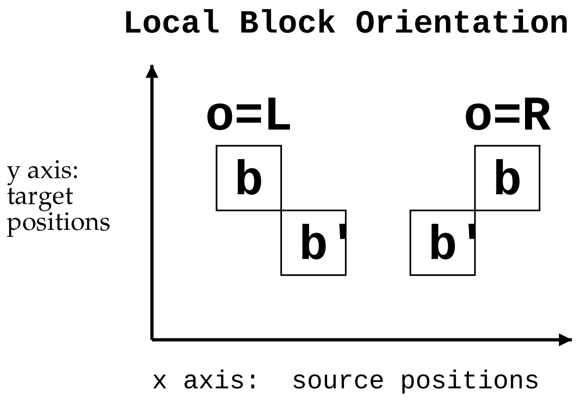


Figure 2: Block b' is the predecessor of block b. The successor block b occurs with either left $o = L$ or right $o = R$ orientation. 'left' and 'right' are defined relative to the x axis ; 'below' is defined relative to the y axis. For some discussion on global re-ordering see Section 6.

$$e_1^N \quad = \quad [e_1^{n_s}]_{s=1}^S = [\,(b_i', o_i, b_i)_{i=1}^{n_s}]_{s=1}^S \qquad (2)$$

Here, n_s is the number of orientation bigrams in the s-th sentence pair. The total number N of orientation bigrams $N = \sum_{s=1}^S n_s$ is about $N = 7.8$ million for our training data consisting of $S = 273\,000$ sentence pairs. The orientation bigram list is used for the parameter training presented in Section 3. Ignoring the bigrams with neutral orientation N reduces the list defined in Eq. 2 to about 5.0 million orientation bigrams. The **Neutral** orientation is handled separately as described in Section 5. Using the reduced orientation bigram list, we collect unigram orientation counts $N_o(b)$: how often a block occurs with a given orientation $o \in \{L, R\}$. $N_L(b) > 0.25 \cdot N_R(b)$ typically holds for blocks b involved in block swapping and the orientation model $p_o(b)$ is defined as:

$$p_o(b) \quad = \quad \frac{N_o(b)}{N_L(b) + N_R(b)}.$$

In order to train a block bigram orientation model as described in Section 3.2, we define a successor set $\delta_s(b')$ for a block b' in the s-th training sentence pair:

$$\delta_s(b') = \{ \text{ number of triples of type } (b', L, b) \text{ or type } (b', R, b) \in e_1^{n_s} \}$$

The successor set $\delta(b')$ is defined for each event in the list e_1^N. The average size of $\delta(b')$ is 1.5 successor blocks. If we were to compute a Viterbi block alignment for a training sentence pair, each block in this block alignment would have at most 1 successor: Blocks may have several successors, because we do not inforce any kind of consistent coverage during training.

During decoding, we generate a list of block orientation bigrams as described above. A DP-based beam search procedure identical to the one used in (Tillmann, 2004) is used to maximize over all oriented block segmentations (b_1^n, o_1^n). During decoding orientation bigrams (b', L, b) with left orientation are only generated if $N_L(b) \geq 3$ for the successor block b.

3 Localized Block Model and Discriminative Training

In this section, we describe the components used to compute the block bigram probability $p(b_i, o_i | b_{i-1}, o_{i-1})$ in Eq. 1. A block orientation pair $(o', b'; o, b)$ is represented as a feature-vector $f(b, o; b', o') \in \mathbb{R}^d$. For a model that uses all the components defined below, d is 6. As feature-vector components, we take the negative logarithm of some block model probabilities. We use the term 'float' feature for these feature-vector components (the model score is stored as a float number). Additionally, we use binary block features. The letters (a)-(f) refer to Table 1:

Unigram Models: we compute (a) the unigram probability $p(b)$ and (b) the orientation probability $p_o(b)$. These probabilities are simple relative frequency estimates based on unigram and unigram orientation counts derived from the data in Eq. 2. For details see (Tillmann, 2004). During decoding, the unigram probability is normalized by the source phrase length.

Two types of Trigram language model: (c) probability of predicting the first target word in the target clump of b_i given the final two words of the target clump of b_{i-1}, (d) probability of predicting the rest of the words in the target clump of b_i. The language model is trained on a separate corpus.

Lexical Weighting: (e) the lexical weight $p(S \mid T)$ of the block $b = (S, T)$ is computed similarly to (Koehn et al., 2003), details are given in Section 3.4.

Binary features: (f) binary features are defined using an indicator function $f(b, b')$ which is 1 if the block pair (b, b') occurs more often than a given threshold N, e.g $N = 2$. Here, the orientation o between the blocks is ignored.

$$f(b, b') = \begin{cases} 1 & N(b, b') > N \\ 0 & \text{else} \end{cases} \qquad (3)$$

3.1 Global Model

In our linear block model, for a given source sentence s, each translation is represented as a sequence of block/orientation pairs $\{b_1^n, o_1^n\}$ consistent with the source. Using features such as those described above, we can parameterize the probability of such a sequence as $P(b_1^n, o_1^n | w, s)$, where w is a vector of unknown model parameters to be estimated from the training data. We use a log-linear probability model and maximum likelihood training— the parameter w is estimated by maximizing the joint likelihood over all sentences. Denote by $\Delta(s)$ the set of possible block/orientation sequences $\{b_1^n, o_1^n\}$ that are consistent with the source sentence s, then a log-linear probability model can be represented as

$$P(b_1^n, o_1^n | w, s) = \frac{\exp(w^T f(b_1^n, o_1^n))}{Z(s)}, \qquad (4)$$

where $f(b_1^n, o_1^n)$ denotes the feature vector of the corresponding block translation, and the partition function is:

$$Z(s) = \sum_{\{b'^m_1, o'^m_1\} \in \Delta(s)} \exp(w^T f(b'^m_1, o'^m_1)).$$

A disadvantage of this approach is that the summation over $\Delta(s)$ can be rather difficult to compute. Consequently some sophisticated approximate inference methods are needed to carry out the computation. A detailed investigation of the global model will be left to another study.

3.2 Local Model Restrictions

In the following, we consider a simplification of the direct global model in Eq. 4. As in (Tillmann, 2004), we model the block bigram probability as $p(b_i, o_i \in \{L, R\} | b_{i-1}, o_{i-1})$ in Eq. 1. We distinguish the two cases (1) $o_i \in \{L, R\}$, and (2) $o_i = N$. Orientation is modeled only in the context of immediate neighbors for blocks that have left or right orientation. The log-linear model is defined as:

$$\begin{aligned} & p(b, o \in \{L, R\} \mid b', o'; w, s) \qquad (5) \\ = & \frac{\exp(w^T f(b, o; b', o'))}{Z(b', o'; s)}, \end{aligned}$$

where s is the source sentence, $f(b, o; b', o')$ is a locally defined feature vector that depends only on the current and the previous oriented blocks (b, o) and (b', o'). The features were described at the beginning of the section. The partition function is given by

$$Z(b', o'; s) = \sum_{(b,o) \in \Delta(b', o'; s)} \exp(w^T f(b, o; b', o')). \quad (6)$$

The set $\Delta(b', o'; s)$ is a restricted set of possible successor oriented blocks that are consistent with the current block position and the source sentence s, to be described in the following paragraph. Note that a straightforward normalization over all block orientation pairs in Eq. 5 is not feasible: there are tens of millions of possible successor blocks b (if we do not impose any restriction). For each block $b = (S, T)$, aligned with a source sentence s, we define a source-induced alternative set:

$$\Gamma(b) = \{ \text{ all blocks } b'' \in \Gamma \text{ that share an identical source phrase with } b \ \}$$

The set $\Gamma(b)$ contains the block b itself and the block target phrases of blocks in that set might differ. To restrict the number of alternatives further, the elements of $\Gamma(b)$ are sorted according to the unigram count $N(b'')$ and we keep at most the top 9 blocks for each source interval s. We also use a modified alternative set $\Gamma_1(b)$, where the block b as well as the elements in the set $\Gamma_1(b)$ are single word blocks. The partition function is computed slightly differently during training and decoding:

Training: for each event (b', o, b) in a sentence pair s in Eq. 2 we compute the successor set $\delta_s(b')$. This defines a set of 'true' block successors. For each true successor b, we compute the alternative set $\Gamma(b)$. $\Delta(b', o'; s)$ is the union of the alternative set for each successor b. Here, the orientation o from the true successor b is assigned to each alternative in $\Gamma(b)$. We obtain on the average 12.8 alternatives per training event (b', o, b) in the list e_1^N.

Decoding: Here, each block b that matches a source interval following b' in the sentence s is a potential successor. We simply set $\Delta(b', o'; s) = \Gamma(b)$. Moreover, setting $Z(b', o'; s) = 0.5$ during decoding does not change performance: the list $\Gamma(b)$ just restricts the possible target translations for a source phrase.

Under this model, the log-probability of a possible translation of a source sentence s, as in Eq. 1, can be written as

$$\begin{aligned} \ln P(b_1^n, o_1^n | w, s) &= \qquad (7) \\ &= \sum_{i=1}^{n} \ln \frac{\exp(w^T f(b_i, o_i; b_{i-1}, o_{i-1}))}{Z(b_{i-1}, o_{i-1}; s)}. \end{aligned}$$

In the maximum-likelihood training, we find w by maximizing the sum of the log-likelihood over observed sentences, each of them has the form in Eq. 7. Although the training methodology is similar to the global formulation given in Eq. 4, this localized version is computationally much easier to manage since the summation in the partition function $Z(b_{i-1}, o_{i-1}; s)$ is now over a relatively small set of candidates. This computational advantage is the main reason that we adopt the local model in this paper.

3.3 Global versus Local Models

Both the global and the localized log-linear models described in this section can be considered as maximum-entropy models, similar to those used in natural language processing, e.g. maximum-entropy models for POS tagging and shallow parsing. In the parsing context, global models such as in Eq. 4 are sometimes referred to as *conditional random field* or CRF (Lafferty et al., 2001).

Although there are some arguments that indicate that this approach has some advantages over localized models such as Eq. 5, the potential improvements are relatively small, at least in NLP applications. For SMT, the difference can be potentially more significant. This is because in our current localized model, successor blocks of different sizes are directly compared to each other, which is intuitively not the best approach (i.e., probabilities of blocks with identical lengths are more comparable). This issue is closely related to the phenomenon of multiple counting of events, which means that a source/target sentence pair can be decomposed into different oriented blocks in our model. In our current training procedure, we select one as the truth, while consider the other (possibly also correct) decisions as non-truth alternatives. In the global modeling, with appropriate normalization, this issue becomes less severe. With this limitation in mind, the localized model proposed here is still an effective approach, as demonstrated by our experiments. Moreover, it is simple both computationally and conceptually. Various issues such as the ones described above can be addressed with more sophisticated modeling techniques, which we shall be left to future studies.

3.4 Lexical Weighting

The lexical weight $p(S \mid T)$ of the block $b = (S, T)$ is computed similarly to (Koehn et al., 2003), but the lexical translation probability $p(s|t)$ is derived from the block set itself rather than from a word alignment, resulting in a simplified training. The lexical weight is computed as follows:

$$\begin{aligned} p(S \mid T) &= \prod_{j=1}^{J} \frac{1}{N_\Gamma(s_j, T)} \sum_{t=1}^{I} p(s_j \mid t_i) \\ p(s_j | t_i) &= \frac{N(b)}{\sum_{b' \in \Gamma_1(b)} N(b')} \end{aligned}$$

Here, the single-word-based translation probability $p(s_j \mid t_i)$ is derived from the block set itself. $b = (s_j, t_i)$ and $b' = (s_j, t_k)$ are single-word blocks, where source and target phrases are of length 1. $N_\Gamma(s_j, t_1^I)$ is the number of blocks $b_k = (s_j, t_k)$ for $k \in 1, \cdots, I$ for which $p(s_j | t_k) > 0.0$.

4 Online Training of Maximum-entropy Model

The local model described in Section 3 leads to the following abstract maximum entropy training formulation:

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{m} \ln \frac{\sum_{j \in \Delta_i} \exp(w^T x_{i,j})}{\exp(w^T x_{i,y_i})}. \quad (8)$$

In this formulation, w is the weight vector which we want to compute. The set Δ_i consists of candidate labels for the i-th training instance, with the true label $y_i \in \Delta_i$. The labels here are block identities , Δ_i corresponds to the alternative set $\Delta(b', o'; s)$ and the 'true' blocks are defined by the successor set $\delta(b')$. The vector $x_{i,j}$ is the feature vector of the i-th instance, corresponding to label $j \in \Delta_i$. The symbol x is short-hand for the feature-vector $f(b, o; b', o')$. This formulation is slightly different from the standard maximum entropy formulation typically encountered in NLP applications, in that we restrict the summation over a subset Δ_i of all labels.

Intuitively, this method favors a weight vector such that for each i, $w^T x_{i,y_i} - w^T x_{i,j}$ is large when $j \neq y_i$. This effect is desirable since it tries to separate the correct classification from the incorrect alternatives. If the problem is completely separable, then it can be shown that the computed linear separator, with appropriate regularization, achieves the largest possible separating margin. The effect is similar to some multi-category generalizations of support vector machines (SVM). However, Eq. 8 is more suitable for non-separable problems (which is often the case for SMT) since it directly models the conditional probability for the candidate labels.

A related method is multi-category perceptron, which explicitly finds a weight vector that separates correct labels from the incorrect ones in a mistake driven fashion (Collins, 2002). The method works by examining one sample at a time, and makes an update $w \rightarrow w + (x_{i,y_i} - x_{i,j})$ when $w^T(x_{i,y_i} - x_{i,j})$ is not positive. To compute the update for a training instance i, one usually pick the j such that $w^T(x_{i,y_i} - x_{i,j})$ is the smallest. It can be shown that if there exist weight vectors that separate the correct label y_i from incorrect labels $j \in \Delta_i$ for all $j \neq y_i$, then the perceptron method can find such a separator. However, it is not entirely clear what this method does when the training data are not completely separable. Moreover, the standard mistake bound justification does not apply when we go through the training data more than once, as typically done in practice. In spite of some issues in its justification, the perceptron algorithm is still very attractive due to its simplicity and computational efficiency. It also works quite well for a number of NLP applications.

In the following, we show that a simple and efficient online training procedure can also be developed for the maximum entropy formulation Eq. 8. The proposed update rule is similar to the perceptron method but with a soft mistake-driven update rule, where the influence of each feature is weighted by the significance of its mistake. The method is essentially a version of the so-called *stochastic gradient descent method*, which has been widely used in complicated stochastic optimization problems such as neural networks. It was argued recently in (Zhang, 2004) that this method also works well for standard convex formulations of binary-classification problems including SVM and logistic regression. Convergence bounds similar to perceptron mistake bounds can be developed, although unlike perceptron, the theory justifies the standard practice of going through the training data more than once. In the non-separable case, the method solves a regularized version of Eq. 8, which has the statistical interpretation of estimating the conditional probability. Consequently, it does not have the potential issues of the perceptron method which we pointed out earlier. Due to the nature of online update, just like perceptron, this method is also very simple to implement and is scalable to large problem size. This is important in the SMT application because we can have a huge number of training instances which we are not able to keep in memory at the same time.

In stochastic gradient descent, we examine one training instance at a time. At the i-th instance, we derive the update rule by maximizing with respect to the term associated with the instance

$$L_i(w) = \ln \frac{\sum_{j \in \Delta_i} \exp(w^T x_{i,j})}{\exp(w^T x_{i,y_i})}$$

in Eq. 8. We do a gradient descent localized to this instance as $w \rightarrow w - \eta_i \frac{\partial}{\partial w} L_i(w)$, where $\eta_i > 0$ is a parameter often referred to as the learning rate. For Eq. 8, the update rule becomes:

$$w \rightarrow w + \eta_i \frac{\sum_{j \in \Delta_i} \exp(w^T x_{i,j})(x_{i,y_i} - x_{i,j})}{\sum_{j \in \Delta_i} \exp(w^T x_{i,j})}. \quad (9)$$

Similar to online algorithms such as the perceptron, we apply this update rule one by one to each training instance (randomly ordered), and may go-through data points repeatedly. Compare Eq. 9 to the perceptron update, there are two main differences, which we discuss below.

The first difference is the weighting scheme. Instead of putting the update weight to a single (most mistaken) feature component, as in the perceptron algorithm, we use a soft-weighting scheme, with each feature component j weighted by a factor $\exp(w^T x_{i,j}) / \sum_{k \in \Delta_i} \exp(w^T x_{i,k})$. A component j with larger $w^T x_{i,j}$ gets more weight. This effect is in principle similar to the perceptron update. The smoothing effect in Eq. 9 is useful for non-separable problems since it does not force an update rule that attempts to separate the data. Each feature component gets a weight that is proportional to its conditional probability.

The second difference is the introduction of a learning rate parameter η_i. For the algorithm to converge, one should pick a decreasing learning rate. In practice, however, it is often more convenient to select a fixed $\eta_i = \eta$ for all i. This leads to an algorithm that approximately solve a regularized version of Eq. 8. If we go through the data repeatedly, one may also decrease the fixed learning rate by monitoring the progress made each time we go through the data. For practical purposes, a fixed small η such as $\eta = 10^{-5}$ is usually sufficient. We typically run forty updates over the training data. Using techniques similar to those of (Zhang, 2004), we can obtain a convergence theorem for our algorithm. Due to the space limitation, we will not present the analysis here.

An advantage of this method over standard maximum entropy training such as GIS (generalized iterative scaling) is that it does not require us to store all the data in memory at once. Moreover, the convergence analysis can be used to show that if m is large, we can get a very good approximate solution by going through the data only once. This desirable property implies that the method is particularly suitable for large scale problems.

5 Experimental Results

The translation system is tested on an Arabic-to-English translation task. The training data comes from the UN news sources. Some punctuation tokenization and some number classing are carried out on the English and the Arabic training data. In this paper, we present results for two test sets: (1) the devtest set uses data provided by LDC, which consists of $1\,043$ sentences with $25\,889$ Arabic words with 4 reference translations. (2) the blind test set is the MT03 Arabic-English DARPA evaluation test set consisting of 663 sentences with $16\,278$ Arabic words with also 4 reference translations. Experimental results are reported in Table 2: here cased BLEU results are reported on MT03 Arabic-English test set (Papineni et al., 2002). The word casing is added as post-processing step using a statistical model (details are omitted here).
In order to speed up the parameter training we filter the original training data according to the two test sets: for each of the test sets we take all the Arabic substrings up to length 12 and filter the parallel training data to include only those training sentence pairs that contain at least one out of these phrases: the 'LDC' training data contains about 273 thousand sentence pairs and the 'MT03' training data contains about 230 thousand sentence pairs. Two block sets are derived for each of the training sets using a phrase-pair selection algorithm similar to (Koehn et al., 2003; Tillmann and Xia, 2003). These block sets also include blocks that occur only once in the training data. Additionally, some heuristic filtering is used to increase phrase translation accuracy (Al-Onaizan et al., 2004).

5.1 Likelihood Training Results

We compare model performance with respect to the number and type of features used as well as with respect to different re-ordering models. Results for 9 experiments are shown in Table 2, where the feature types are described in Table 1. The first 5 experimental results are obtained by carrying out the likelihood training described in Section 3. Line 1 in Table 2 shows the performance of the baseline block unigram '*MON*' model which uses two 'float' features: the unigram probability and the boundary-word language model probability. No block re-ordering is allowed for the baseline model (a **monotone** block sequence is generated). The '*SWAP*' model in line 2 uses the same two features, but neighbor blocks can be swapped. No performance increase is obtained for this model. The '*SWAP* & *OR*' model uses an orientation model as described in Section 3. Here, we obtain a small but significant improvement over the baseline model. Line 4 shows that by including two additional 'float' features: the lexical weighting and the language model probability of predicting the second and subsequent words of the target clump yields a further significant improvement. Line 5 shows that including binary features and training their weights on the training data actually decreases performance. This issue is addressed in Section 5.2.

The training is carried out as follows: the results in line 1-4 are obtained by training 'float' weights only. Here, the training is carried out by running only once over 10 % of the training data. The model including the binary features is trained on the entire training data. We obtain about 3.37 million features of the type defined in Eq. 3 by setting the threshold $N = 3$. Forty iterations over the training data take about 2 hours on a single Intel machine. Although the online algorithm does not require us to do so, our training procedure keeps the entire training data and the weight vector w in about 2 gigabytes of memory.

For blocks with neutral orientation $o = N$, we train a separate model that does not use the orientation model feature or the binary features. E.g. for the results in line 5 in Table 2, the neutral model would use the features $(a), (c), (d), (e)$, but not (b) and (f). Here, the neutral model is trained on the neutral orientation bigram subsequence that is part of Eq. 2.

5.2 Modified Weight Training

We implemented the following variation of the likelihood training procedure described in Section 3, where we make use of the 'LDC' devtest set. First, we train a model on the 'LDC' training data using 5 float features and the binary features. We use this model to decode

Table 1: List of feature-vector components. For a description, see Section 3.

Description
(a) Unigram probability
(b) Orientation probability
(c) LM first word probability
(d) LM second and following words probability
(e) Lexical weighting
(f) Binary Block Bigram Features

Table 2: Cased BLEU translation results with confidence intervals on the MT03 test data. The third column summarizes the model variations. The results in lines 8 and 9 are for a cheating experiment: the float weights are trained on the test data itself.

	Re-ordering	Components	BLEU
1	’*MON*’	(a),(c)	32.3 ± 1.5
2	’*SWAP*’	(a),(c)	32.3 ± 1.5
3	’*SWAP* & *OR*’	(a),(b),(c)	33.9 ± 1.4
4	’*SWAP* & *OR*’	(a)-(e)	37.7 ± 1.5
5	’*SWAP* & *OR*’	(a)-(f)	37.2 ± 1.6
6	’*SWAP* & *OR*’	(a)-(e) (ldc devtest)	37.8 ± 1.5
7	’*SWAP* & *OR*’	(a)-(f) (ldc devtest)	38.2 ± 1.5
8	’*SWAP* & *OR*’	(a)-(e) (mt03 test)	39.0 ± 1.5
9	’*SWAP* & *OR*’	(a)-(f) (mt03 test)	39.3 ± 1.6

the devtest ’LDC’ set. During decoding, we generate a ’translation graph’ for every input sentence using a procedure similar to (Ueffing et al., 2002): a translation graph is a compact way of representing candidate translations which are close in terms of likelihood. From the translation graph, we obtain the $1\,000$ best translations according to the translation score. Out of this list, we find the block sequence that generated the top BLEU-scoring target translation. Computing the top BLEU-scoring block sequence for all the input sentences we obtain:

$$e_1^{N'} \quad = \quad [\,(b_i', o_i, b_i)_{i=1}^{n_{s'}}]_1^{S'} \ , \tag{10}$$

where $N' \approx 9400$. Here, N' is the number of blocks needed to decode the entire devtest set. Alternatives for each of the events in $e_1^{N'}$ are generated as described in Section 3.2. The set of alternatives is further restricted by using only those blocks that occur in some translation in the $1\,000$-best list. The 5 float weights are trained on the modified training data in Eq. 10, where the training takes only a few seconds. We then decode the ’MT03’ test set using the modified ’float’ weights. As shown in line 4 and line 6 there is almost no change in performance between training on the original training data in Eq. 2 or on the modified training data in Eq. 10. Line 8 shows that even when training the float weights on an event set obtained from the test data itself in a cheating experiment, we obtain only a moderate performance improvement from 37.7 to 39.0. For the experimental results in line 7 and 9, we use the same five float weights as trained for the experiments in line 6 and 8 and keep them fixed while training the binary feature weights only. Using the binary features leads to only a minor improvement in BLEU from 37.8 to 38.2 in line 7. For this best model, we obtain a 18.6 % BLEU improvement over the baseline.

From our experimental results, we draw the following conclusions: (1) the translation performance is largely dominated by the ’float’ features, (2) using the same set of ’float’ features, the performance doesn’t change much when training on training, devtest, or even test data. Although, we do not obtain a significant improvement from the use of binary features, currently, we expect the use of binary features to be a promising approach for the following reasons:

- The current training does not take into account the block interaction on the sentence level. A more accurate approximation of the global model as discussed in Section 3.1 might improve performance.
- As described in Section 3.2 and Section 5.2, for efficiency reasons alternatives are computed from source phrase matches only. During training, more accurate local approximations for the partition function in Eq. 6 can be obtained by looking at block translations in the context of translation sequences. This involves the computationally expensive generation of a translation graph for each training sentence pair. This is future work.
- As mentioned in Section 1, viewing the translation process as a sequence of local discussions makes it similar to other NLP problems such as POS tagging, phrase chunking, and also statistical parsing. This similarity may facilitate the incorporation of these approaches into our translation model.

6 Discussion and Future Work

In this paper we proposed a method for discriminatively training the parameters of a block SMT decoder. We discussed two possible approaches: global versus local. This work focused on the latter, due to its computational advantages. Some limitations of our approach have also been pointed out, although our experiments showed that this simple method can significantly improve the baseline model.

As far as the log-linear combination of float features is concerned, similar training procedures have been proposed in (Och, 2003). This paper reports the use of 8

features whose parameter are trained to optimize performance in terms of different evaluation criteria, e.g. BLEU. On the contrary, our paper shows that a significant improvement can also be obtained using a likelihood training criterion.

Our modified training procedure is related to the discriminative re-ranking procedure presented in (Shen et al., 2004). In fact, one may view discriminative reranking as a simplification of the global model we discussed, in that it restricts the number of candidate global translations to make the computation more manageable. However, the number of possible translations is often exponential in the sentence length, while the number of candidates in a typically reranking approach is fixed. Unless one employs an elaborated procedure, the candidate translations may also be very similar to one another, and thus do not give a good coverage of representative translations. Therefore the reranking approach may have some severe limitations which need to be addressed. For this reason, we think that a more principled treatment of global modeling can potentially lead to further performance improvements.

For future work, our training technique may be used to train models that handle global sentence-level reorderings. This might be achieved by introducing orientation sequences over phrase types that have been used in ((Schafer and Yarowsky, 2003)). To incorporate syntactic knowledge into the block-based model, we will examine the use of additional real-valued or binary features, e.g. features that look at whether the block phrases cross syntactic boundaries. This can be done with only minor modifications to our training method.

Acknowledgment

This work was partially supported by DARPA and monitored by SPAWAR under contract No. N66001-99-2-8916. The paper has greatly profited from suggestions by the anonymous reviewers.

References

Yaser Al-Onaizan, Niyu Ge, Young-Suk Lee, Kishore Papineni, Fei Xia, and Christoph Tillmann. 2004. IBM Site Report. In *NIST 2004 Machine Translation Workshop*, Alexandria, VA, June.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP'02*.

Philipp Koehn, Franz-Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of the HLT-NAACL 2003 conference*, pages 127–133, Edmonton, Canada, May.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-01*, pages 282–289.

Franz-Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved Alignment Models for Statistical Machine Translation. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC 99)*, pages 20–28, College Park, MD, June.

Och et al. 2004. A Smorgasbord of Features for Statistical Machine Translation. In *Proceedings of the Joint HLT and NAACL Conference (HLT 04)*, pages 161–168, Boston, MA, May.

Franz-Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of the 41st Annual Conf. of the Association for Computational Linguistics (ACL 03)*, 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 the 40th Annual Conf. of the Association for Computational Linguistics (ACL 02)*, pages 311–318, Philadelphia, PA, July.

Charles Schafer and David Yarowsky. 2003. Statistical Machine Translation Using Coercive Two-Level Syntactic Translation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP 03)*, pages 9–16, Sapporo, Japan, July.

Libin Shen, Anoop Sarkar, and Franz-Josef Och. 2004. Discriminative Reranking of Machine Translation. In *Proceedings of the Joint HLT and NAACL Conference (HLT 04)*, pages 177–184, Boston, MA, May.

Christoph Tillmann and Fei Xia. 2003. A Phrase-based Unigram Model for Statistical Machine Translation. In *Companian Vol. of the Joint HLT and NAACL Conference (HLT 03)*, pages 106–108, Edmonton, Canada, June.

Christoph Tillmann. 2004. A Unigram Orientation Model for Statistical Machine Translation. In *Companian Vol. of the Joint HLT and NAACL Conference (HLT 04)*, pages 101–104, Boston, MA, May.

Nicola Ueffing, Franz-Josef Och, and Hermann Ney. 2002. Generation of Word Graphs in Statistical Machine Translation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP 02)*, pages 156–163, Philadelphia, PA, July.

Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 04*, pages 919–926.

Instance-based Sentence Boundary Determination by Optimization for Natural Language Generation

Shimei Pan and **James C. Shaw**
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
{shimei,shawjc}@us.ibm.com

Abstract

This paper describes a novel instance-based sentence boundary determination method for natural language generation that optimizes a set of criteria based on examples in a corpus. Compared to existing sentence boundary determination approaches, our work offers three significant contributions. First, our approach provides a general domain independent framework that effectively addresses sentence boundary determination by balancing a comprehensive set of sentence complexity and quality related constraints. Second, our approach can simulate the characteristics and the style of naturally occurring sentences in an application domain since our solutions are optimized based on their similarities to examples in a corpus. Third, our approach can adapt easily to suit a natural language generation system's capability by balancing the strengths and weaknesses of its subcomponents (e.g. its aggregation and referring expression generation capability). Our final evaluation shows that the proposed method results in significantly better sentence generation outcomes than a widely adopted approach.

1 Introduction

The problem of sentence boundary determination in natural language generation exists when more than one sentence is needed to convey multiple concepts and propositions. In the classic natural language generation (NLG) architecture (Reiter, 1994), sentence boundary decisions are made during the sentence planning stage in which the syntactic structure and wording of sentences are decided. Sentence boundary determination is a complex process that directly impacts a sentence's readability (Gunning, 1952), its semantic cohesion, its syntactic and lexical realizability, and its smoothness between sentence transitions. Sentences that are too complex are hard to understand, so are sentences lacking semantic cohesion and cross-sentence coherence. Further more, bad sentence boundary decisions may even make sentences unrealizable.

To design a sentence boundary determination method that addresses these issues, we employ an instance-based approach (Varges and Mellish, 2001; Pan and Shaw, 2004). Because we optimize our solutions based on examples in a corpus, the output sentences can demonstrate properties, such as similar sentence length distribution and semantic grouping similar to those in the corpus. Our approach also avoids problematic sentence boundaries by optimizing the solutions using all the instances in the corpus. By taking a sentence's lexical and syntactic realizability into consideration, it can also avoid sentence realization failures caused by bad sentence boundary decisions. Moreover, since our solution can be adapted easily to suit the capability of a natural language generator, we can easily tune the algorithm to maximize the generation quality. To the best of our knowledge, there is no existing comprehensive solution that is domain-independent and possesses all the above qualities. In summary, our work offers three significant contributions:

1. It provides a general and flexible sentence

Proceedings of the 43rd Annual Meeting of the ACL, pages 565–572,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

boundary determination framework which takes a comprehensive set of sentence complexity and quality related criteria into consideration and ensures that the proposed algorithm is sensitive to not only the complexity of the generated sentences, but also their semantic cohesion, multi-sentence coherence and syntactic and lexical realizability.

2. Since we employ an instance-based method, the proposed solution is sensitive to the style of the sentences in the application domain in which the corpus is collected.
3. Our approach can be adjusted easily to suit a sentence generation system's capability and avoid some of its known weaknesses.

Currently, our work is embodied in a multimodal conversation application in the real-estate domain in which potential home buyers interact with the system using multiple modalities, such as speech and gesture, to request residential real-estate information (Zhou and Pan, 2001; Zhou and Chen, 2003; Zhou and Aggarwal, 2004). After interpreting the request, the system formulates a multimedia presentation, including automatically generated speech and graphics, as the response (Zhou and Aggarwal, 2004). The proposed sentence boundary determination module takes a set of propositions selected by a content planner and passes the sentence boundary decisions to SEGUE (Pan and Shaw, 2004), an instance-based sentence generator, to formulate the final sentences. For example, our system is called upon to generate responses to a user's request: "Tell me more about this house." Even though not all of the main attributes of a house (more than 20) will be conveyed, it is clear that a good sentence boundary determination module can greatly ease the generation process and improve the quality of the output.

In the rest of the paper, we start with a discussion of related work, and then describe our instance-base approach to sentence boundary determination. Finally, we present our evaluation results.

2 Related Work

Existing approaches to sentence boundary determination typically employ one of the following strategies. The first strategy uses domain-specific heuristics to decide which propositions can be combined. For example, *Proteus* (Davey, 1979; Ritchie, 1984) produces game descriptions by employing domain-specific sentence scope heuristics. This approach can work well for a particular application, however, it is not readily reusable for new applications.

The second strategy is to employ syntactic, lexical, and sentence complexity constraints to control the aggregation of multiple propositions (Robin, 1994; Shaw, 1998). These strategies can generate fluent complex sentences, but they do not take other criteria into consideration, such as semantic cohesion. Further more, since these approaches do not employ global optimization as we do, the content of each sentence might not be distributed evenly. This may cause dangling sentence problem (Wilkinson, 1995).

Another strategy described in Mann and Moore(1981) guided the aggregation process by using an evaluation score that is sensitive to the structure and term usage of a sentence. Similar to our approach, they rely on search to find an optimal solution. The main difference between this approach and ours is that their evaluation score is computed based on preference heuristics. For example, all the semantic groups existing in a domain have to be coded specifically in order to handle semantic grouping. In contrast, in our framework, the score is computed based on a sentence's similarity to corpus instances, which takes advantage of the naturally occurring semantic grouping in the corpus.

Recently, Walker (2002) and Stent (2004) used statistical features derived from corpus to rank generated sentence plans. Because the plan ranker was trained with existing examples, it can choose a plan that is consistent with the examples. However, depending on the features used and the size of the training examples, it is unclear how well it can capture patterns like semantic grouping and avoid problems likes dangling sentences.

3 Examples

Before we describe our approach in detail, we start with a few examples from the real-estate domain to demonstrate the properties of the proposed approach.

First, sentence complexity impacts sentence boundary determination. As shown in Table 1, after receiving a user's request (U1) for the details of a house, the content planner asked the sentence planner to describe the house with a set of attributes including its asking price, style, number of bedrooms, number of bathrooms, square footage, garage, lot size, property tax, and its associated town and school

Example	Turn	Sentence
E1	U1	Tell me more about this house
	S1	This is a 1 million dollar 3 bedroom, 2 bathroom, 2000 square foot colonial with 2 acre of land, 2 car garage, annual taxes 8000 dollars in Armonk and in the Byram Hills school district.
	S2	This is a 1 million dollar house. This is a 3 bedroom house. This is a 2 bathroom house. This house has 2000 square feet. This house has 2 acres of land. This house has 2 car garage. This is a colonial house. The annual taxes are 8000 dollars. This house is in Armonk. This house is in the Byram Hills school district.
	S3	This is a 3 bedroom, 2 bathroom, 2000 square foot colonial located in Armonk with 2 acres of land. The asking price is 1 million dollar and the annual taxes are 8000 dollars. The house is located in the Byram Hills School District.
E2	S4	This is a 1 million dollar 3 bedroom house. This is a 2 bathroom house with annual taxes of 8000 dollars.
	S5	This is a 3 bedroom and 2 bathroom house. Its price is 1 million dollar and its annual taxes are 8000 dollars.
E3	S6	The tax rate of the house is 3 percent.
	S7	The house has an asphalt roof.
E4	S8	This is a 3 bedroom, 2 bathroom colonial with 2000 square feet and 2 acres of land.
	S9	The house has 2 bedrooms and 3 bathrooms. This house is a colonial. It has 2000 square feet. The house is on 2 acres of land.

Table 1: Examples

district name. Without proper sentence boundary determination, a sentence planner may formulate a single sentence to convey all the information, as in S1. Even though S1 is grammatically correct, it is too complex and too exhausting to read. Similarly, output like S2, despite its grammatical correctness, is choppy and too tedious to read. In contrast, our instance-based sentence boundary determination module will use examples in a corpus to partition those attributes into several sentences in a more balanced manner (S3).

Semantic cohesion also influences the quality of output sentences. For example, in the real-estate domain, the number of bedrooms and number of bathrooms are two closely related concepts. Based on our corpus, when both concepts appear, they almost always conveyed together in the same sentence. Given this, if the content planner wants to convey a house with the following attributes: price, number of bedrooms, number of bathrooms, and property tax, S4 is a less desirable solution than S5 because it splits these concepts into two separate sentences. Since we use instance-based sentence boundary determination, our method generates S5 to minimize the difference from the corpus instances.

Sentence boundary placement is also sensitive to the syntactic and lexical realizability of grouped items. For example, if the sentence planner asks the surface realizer to convey two propositions S6 and S7 together in a sentence, a realization failure will be triggered because both S6 and S7 only exist in the corpus as independent sentences. Since neither of them can be transformed into a modifier based on the corpus, S6 and S7 cannot be aggregated in our system. Our method takes a sentence's lexical and syntactic realizability into consideration in order to avoid making such aggregation request to the surface realizer in the first place.

A generation system's own capability may also influence sentence boundary determination. Good sentence boundary decisions will balance a system's strengths and weaknesses. In contrast, bad decisions will expose a system's venerability. For example, if a sentence generator is good at performing aggregations and weak on referring expressions, we may avoid incoherence between sentences by preferring aggregating more attributes in one sentence (like in S8) rather than by splitting them into multiple sentences (like in S9).

In the following, we will demonstrate how our approach can achieve all the above goals in a unified instance-based framework.

4 Instance-based boundary determination

Instance-based generation automatically creates sentences that are similar to those generated by humans, including their way of grouping semantic content, their wording and their style. Previously, Pan and Shaw (2004) have demonstrated that instance-based learning can be applied successfully in generating new sentences by piecing together existing words and segments in a corpus. Here, we want to demonstrate that by applying the same principle, we can make better sentence boundary decisions.

The key idea behind the new approach is to find a sentence boundary solution that minimizes the expected difference between the sentences resulting from these boundary decisions and the examples in the corpus. Here we measure the expected difference based a set of cost functions.

4.1 Optimization Criteria

We use three sentence complexity and quality related cost functions as the optimization criteria: sentence boundary cost, insertion cost and deletion cost.

Sentence boundary cost (SBC): Assuming P is a set of propositions to be conveyed and S is a collection of example sentences selected from the corpus to convey P. Then we say P can be realized by S with a sentence boundary cost that is equal to $(|S|-1)*SBC$ in which $|S|$ is the number of sentences and SBC is the sentence boundary cost. To use a specific example from the real-estate domain, the input P has three propositions:

p_1. House1 has-attr (style=colonial).

p_2. House1 has-attr(bedroom=3).

p_3. House1 has-attr(bathroom=2).

One solution, S, contains 2 sentences:

s_1. This is a 3 bedroom, 2 bathroom house.

s_2. This is a colonial house.

Since only one sentence boundary is involved, S is a solution containing one boundary cost. In the above example, even though both s_1 and s_2 are grammatical sentences, the transition from s_1 to s_2 is not quite smooth. They sound choppy and disjointed. To penalize this, whenever there is a sentence break, there is a SBC. In general, the SBC is a parameter that is sensitive to a generation system's capability such as its competence in reference expression generation. If a generation system does not have a robust approach for tracking the focus across sentences, it is likely to be weak in referring expression generation and adding sentence boundaries are likely to cause fluency problems. In contrast, if a generation system is very capable in maintaining the coherence between sentences, the proper sentence boundary cost would be lower.

Insertion cost: Assume P is the set of propositions to be conveyed, and C_i is an instance in the corpus that can be used to realize P by inserting a missing proposition p_j to C_i, then we say P can be realized using C_i with an insertion cost of $icost(C_H, p_j)$, in which C_H is the host sentence in the corpus containing proposition p_j. Using an example from our real-estate domain, assume the input P=(p_2, p_3, p_4), where

p_4. House1 has-attr (square footage=2000).

Assume C_i is a sentence selected from the corpus to realize P: "This is 3 bedroom 2 bathroom house". Since C_i does not contain p_4, p_4 needs to be added. We say that P can be realized using C_i by inserting a proposition p_4 with an insertion cost of $icost(C_H, p_4)$, in which C_H is a sentence in the corpus such as "This is a house with 2000 square feet."

The insertion cost is influenced by two main factors: the syntactic and lexical insertability of the proposition p_j and a system's capability in aggregating propositions. For example, if in the corpus, the proposition p_j is always realized as an independent sentence and never as a modifier, $icost(*, p_j)$ should be extremely high, which effectively prohibit p_j from becoming a part of another sentence. $icost(*, p_j)$ is defined as the minimum insertion cost among all the $icost(C_H, p_j)$. Currently $icost(C_H, p_j)$ is computed dynamically based on properties of corpus instances. In addition, since whether a proposition is insertable depends on how capable an aggregation module can combine propositions correctly into a sentence, the insertion cost should be assigned high or low accordingly.

Deletion cost: Assume P is a set of input propositions to be conveyed and C_i is an instance in the corpus that can be used to convey P by deleting an unneeded proposition p_j in C_i. Then, we say P can be realized using C_i with a deletion cost $dcost(C_i, p_j)$. As a specific example, assuming the input is P=(p_2, p_3, p_4), C_i is an instance in the corpus "This is a 3 bedroom, 2 bathroom, 2000 square foot colonial house." In addition to the propositions p_2, p_3 and p_4, C_i also conveys a proposition p_1. Since p_1 is not needed when conveying P, we say that P can be realized using C_i by deleting proposition p_1 with a deletion cost of $dcost(C_i, p_1)$.

The deletion cost is affected by two main factors as well: first the syntactic relation between p_j and its host sentence. Given a new instance C_i, "This 2000 square foot 3 bedroom, 2 bathroom house is a colonial", deleting p_1, the main object of the verb, will make the rest of the sentence incomplete. As a result, $dcost(C_i, p_1)$ is very expensive. In contrast, $dcost(C_i, p_4)$ is low because the resulting sentence is still grammatically sound. Currently $dcost(C_i, p_j)$ is computed dynamically based on properties of corpus instances. Second, the expected performance of a generation system in deletion also impacts the deletion cost. Depending on the sophistication of the generator to handle various deletion situations, the expected deletion cost can be high if the method employed is naive and error prone, or is low if the system can handle most cases accurately.

Overall cost: Assume P is the set of propositions to be conveyed and S is the set of instances in the corpus that are chosen to realize P by applying a set of insertion, deletion and sentence breaking operations, the overall cost of the solution

$$\begin{aligned} Cost(P) &= \sum_{C_i}(W_i * \sum_j icost(C_{Hj}, p_j) \\ &\quad + W_d * \sum_k dcost(C_i, p_k)) \\ &\quad + (N_b - 1) * SBC \end{aligned}$$

in which W_i, W_d and SBC are the insertion weight, deletion weight and sentence boundary cost; N_b is the number of sentences in the solution, C_i is a corpus instance been selected to construct the solution and C_{Hj} is the host sentence that proposition p_j belongs.

4.2 Algorithm: Optimization based on overall cost

We model the sentence boundary determination process as a branch and bound tree search problem. Before we explain the algorithm itself, first a few notations. The input P is a set of input propositions chosen by the content planner to be realized. Σ is the set of all possible propositions in an application domain. Each instance C_i in the corpus C is represented as a subset of Σ. Assume S is a solution to P, then it can be represented as the overall cost plus a list of pairs like (C_is, O_is), in which C_is is one of the instances selected to be used in that solution, O_is is a set of deletion, insertion operations that can be applied to C_is to transform it to a subsolution S_i. To explain this representation further, we use a specific example in which P=(a, d, e, f), Σ=(a, b, c, d, e, f g, h, i). One of the boundary solution S can be represented as

$$\begin{aligned} S &= (Cost(S), (S1, S2)) \\ S_1 &= (C_1 = (a, b, d, i), delete(b, i)), \\ S_2 &= (C_2 = (e), insert(f \text{ as in } C_3 = (f, g))) \\ Cost(S) &= W_d * (dcost(C_1, b) + dcost(C_1, i)) + \\ &\quad W_i * icost(C_3, f) + 1 * SBC \end{aligned}$$

in which C_1 and C_2 are two corpus instances selected as the bases to formulate the solution and C_3 is the host sentence containing proposition f.

The general idea behind the instance-based branch and bound tree search algorithm is that given an input, P, for each corpus instance C_i, we construct a search branch, representing all possible ways to realize the input using the instance plus deletions, insertions and sentence breaks. Since each sentence break triggers a recursive call to our sentence boundary determination algorithm, the complexity of the algorithm is NP-hard. To speed up the process, for each iteration, we prune unproductive branches using an upper bound derived by several greedy algorithms. The details of our sentence boundary determination algorithm, $sbd(P)$, are described below. P is the set of input propositions.

1. Set the current upper bound, UB, to the minimum cost of solutions derived by greedy algorithms, which we will describe later. This value is used to prune unneeded branches to make the search more efficient.

2. For each instance C_i in corpus C in which $(C_i \cap P) \neq \emptyset$, loop from step 3 to 9. The goal here is to identify all the useful corpus instances for realizing P.

3. Delete all the propositions $p_j \in D$ in which $D = C_i - P$ (D contains propositions in C_i but not exist in P) with cost $Cost_d(P) = W_d * \sum_{P_j \in D} dcost(C_i, p_j)$. This step computes the deletion operators and their associated costs.

4. Let $I = P - C_i$ (I contains propositions in P but not in C_i). For each subset $E_j \subseteq I$ (E_j includes $\emptyset$ and I itself), iterate through step 5 to 9. These steps figure out all the possible ways to add the missing propositions, including inserting into the instance C_i and separating the rest as independent sentence(s).

5. Generate a solution in which $\forall p_k \in E_j$, insert p_k to C_i. All the propositions in $Q = I - E_j$ will be realized in different sentences, thus incurring a SBC.

6. We update the cost $Cost(P)$ to

$$Cost_d(P) + W_i * \sum_{p_k \in E_j} icost(*, p_k) + SBC + Cost(Q)$$

   in which $Cost(Q)$ is the cost of sbd(Q) which recursively computes the best solution for input Q and $Q \subset P$. To facilitate dynamic programming, we remember the best solution for Q derived by sbd(Q) in case Q is used to formulate other solutions.

7. If the lower bound for Cost(P) is greater than the established upper bound UB, prune this branch.

8. Using the notation described in the beginning of Sec. 4.2, we update the current solution to

$$\begin{aligned} sbd(P) &= (Cost(P), (C_i, delete_{\forall p_j \in D}(p_j), \\ &\quad insert_{\forall p_k \in E_j}(p_k))) \bigoplus sbd(Q) \end{aligned}$$

   in which $\bigoplus$ is an operator that composes two partial solutions.

9. If sbd(P) is a complete solution (either Q is empty or have a known best solution) and $Cost(P) < UB$, update the upper bound $UB = Cost(P)$.

10. Output the solution with the lowest overall cost.

To establish the initial UB for pruning, we use the minimum of the following three bounds. In general, the tighter the UB is, the more effective the pruning is.

Greedy set partition: we employ a greedy set partition algorithm in which we first match the set $S \subset P$ with the largest $|S|$. Repeat the same process for P' where $P' = P - S$. The solution cost is $Cost(P) = (N - 1) * SBC$, and N is the number of sentences in the solution. The complexity of this computation is $O(|P|)$, where $|P|$ is the number of propositions in P.

Revised minimum set covering: we employ a greedy minimum set covering algorithm in which we first find the set S in the corpus that maximizes the overlapping of propositions in the input P. The unwanted propositions in $S - P$ are deleted. Assume $P' = P - S$, repeat the same process to P' until P' is empty. The only difference between this and the previous approach is that S here might not be a subset of P. The complexity of this computation is $O(|P|)$.

One maximum overlapping sentence: we first identify the instance C_i in corpus that covers the maximum number of propositions in P. To arrive at a solution for P, the rest of the propositions not covered by C_i are inserted into C_i and all the unwanted propositions in C_i are deleted. The cost of this solution is

$$W_d * \sum_{p_j \in D} dcost(C_i, p_j) + W_i * \sum_{p_k \in I} icost(*, p_k)$$

in which D includes proposition in C_i but not in P, and I includes propositions in P but not in C_i.

Currently, we update UB only after a complete solution is found. It is possible to derive better UB by establishing the upper bound for each partial solution, but the computational overhead might not justify doing so.

4.3 Approximation Algorithm

Even with pruning and dynamic programming, the exact solution still is very expensive computationally. Computing exact solution for an input size of 12 propositions has over 1.6 millions states and takes more than 30 minutes (see Figure 1). To make the search more efficient for tasks with a large number of propositions in the input, we naturally seek a greedy strategy in which at every iteration the algorithm myopically chooses the next best step without regard for its implications on future moves. One greedy search policy we implemented explores the branch that uses the instance with maximum overlapping propositions with the input and ignores all branches exploring other corpus instances. The intuition behind this policy is that the more overlap an instance has with the input, the less insertions or sentence breaks are needed.

Figure 1 and Figure 2 demonstrate the tradeoff between computation efficiency and accuracy. In this graph, we use instances from the real-estate corpus with size 250, we vary the input sentence length from one to twenty and the results shown in the graphs are average value over several typical weight configurations ((W_d,W_i,SBC)=

(1,3,5),(1,3,7),(1,5,3),(1,7,3),(1,1,1)). Figure 2 compares the quality of the solutions when using exact solutions versus approximation. In our interactive multimedia system, we currently use exact solution for input size of 7 propositions or less and switch to greedy for any larger input size to ensure sub-second performance for the NLG component.

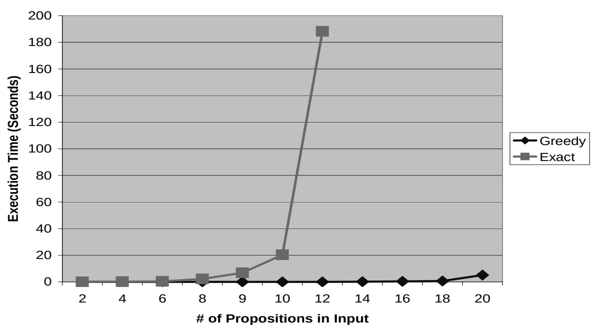


Figure 1: Speed difference between exact solutions and approximations

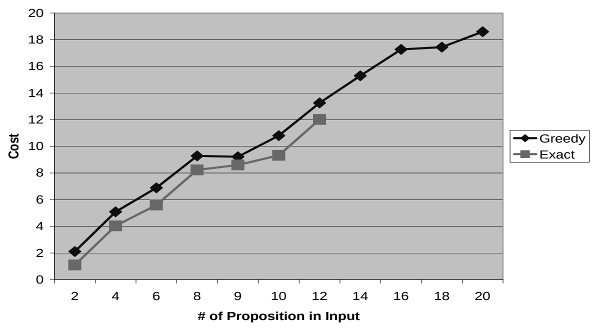


Figure 2: Cost difference between exact solutions and approximations

Measures	Ours	B-3	B-6
Dangling sentence (7)	0	100%	100%
Split Semantic Group	1%	61%	21%
Realization Failure	0	56%	72%
Fluency	59%	4%	8%

Table 2: Comparisons

5 Evaluations

To evaluate the quality of our sentence boundary decisions, we implemented a baseline system in which boundary determination of the aggregation module is based on a threshold of the maximum number of propositions allowed in a sentence (a simplified version of the second strategy in Section 2. We have tested two threshold values, the average (3) and maximum (6) number of propositions among corpus instances. Other sentence complexity measures, such as the number of words and depth of embedding are not easily applicable for our comparison because they require the propositions to be realized first before the boundary decisions can be made.

We tune the relative weight of our approach to best fit our system's capability. Currently, the weights are empirically established to $W_d = 1$, $W_i = 3$ and $SBC = 3$. Based on the output generated from both systems, we derive four evaluation metrics:

1. **Dangling sentences**: We define dangling sentences as the short sentences with only one proposition that follow long sentences. This measure is used to verify our claim that because we use global instead of local optimization, we can avoid generating dangling sentences by making more balanced sentence boundary decisions. In contrast, the baseline approaches have dangling sentence problem when the input proposition is 1 over the multiple of the threshold values. The first row of Table 2 shows that when the input proposition length is set to 7, a pathological case, among the 200 input proposition sets randomly generated, the baseline approach always produce dangling sentences (100%). In contrast, our approach always generates more balanced sentences (0%).

2. **Semantic group splitting**. Since we use an instance-based approach, we can maintain the semantic cohesion better. To test this, we randomly generated 200 inputs with up to 10 propositions containing semantic grouping of both the number of bedrooms and number of bathrooms. The second row, Split Semantic Group, in Table 2 shows that our algorithm can maintain semantic group much better than the baseline approach. Only in 1% of the output sentences, our algorithm generated number of bedrooms and number of bathrooms in separate sentences. In contrast, the baseline approaches did much worse (61% and 21%).

3. **Sentence realization failure**. This measure is used to verify that since we also take a sentence's lexical and syntactical realizability into consideration, our sentence boundary decisions will result in less sentence realization failures.

An realization failure occurs when the aggregation module failed to realize one sentence for all the propositions grouped by the sentence boundary determination module. The third row in Table 2, Realization Failure, indicates that given 200 randomly generated input proposition sets with length from 1 to 10, how many realization happened in the output. Our approach did not have any realization failure while for the baseline approaches, there are 56% and 72% outputs have one or more realization failures.

4. **Fluency**. This measure is used to verify our claim that since we also optimize our solutions based on boundary cost, we can reduce incoherence across multiple sentences. Given 200 randomly generated input propositions with length from 1 to 10, we did a blind test and presented pairs of generated sentences to two human subjects randomly and asked them to rate which output is more coherent. The last row, Fluency, in Table 2 shows how often the human subjects believe that a particular algorithm generated better sentences. The output of our algorithm is preferred for more than 59% of the cases, while the baseline approaches are preferred 4% and 8%, respectively. The other percentages not accounted for are cases where the human subject felt there is no significant difference in fluency between the two given choices. The result from this evaluation clearly demonstrates the superiority of our approach in generating coherent sentences.

6 Conclusion

In the paper, we proposed a novel domain independent instance-based sentence boundary determination algorithm that is capable of balancing a comprehensive set of generation capability, sentence complexity, and quality related constraints. This is the first domain-independent algorithm that possesses many desirable properties, including balancing a system's generation capabilities, maintaining semantic cohesion and cross sentence coherence, and preventing severe syntactic and lexical realization failures. Our evaluation results also demonstrate the superiority of the approach over a representative domain independent sentence boundary solution.

References

Anthony C. Davey. 1979. *Discourse Production*. Edinburgh University Press, Edinburgh.

Robert Gunning. 1952. *The Technique of Clear Writing*. McGraw-Hill.

William C. Mann and James A. Moore. 1981. Computer generation of multiparagraph English text. *American Journal of Computational Linguistics*, 7(1):17–29.

Shimei Pan and James Shaw. 2004. SEGUE: A hybrid case-based surface natural language generator. In *Proc. of ICNLG*, Brockenhurst, U.K.

Ehud Reiter. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proc. of INLG*, Kennebunkport, Maine.

Graeme D. Ritchie. 1984. A rational reconstruction of the Proteus sentence planner. In *Proc. of the COLING and the ACL*, Stanford, CA.

Jacques Robin. 1994. Automatic generation and revision of natural language summaries providing historical background. In *Proc. of the Brazilian Symposium on Artificial Intelligence*, Fortaleza, CE, Brazil.

James Shaw. 1998. Segregatory coordination and ellipsis in text generation. In *Proc. of the COLING and the ACL.*, Montreal, Canada.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proc. of the ACL*, Barcelona, Spain.

Sebastian Varges and Chris Mellish. 2001. Instance-based natural language generation. In *Proc. of the NAACL*, Pittsburgh, PA.

Marilyn Walker, Owen Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language*.

John Wilkinson. 1995. Aggregation in natural language generation: Another look. Co-op work term report, Dept. of Computer Science, University of Waterloo.

Michelle Zhou and Vikram Aggarwal. 2004. An optimization-based approach to dynamic data content selection in intelligent multimedia interfaces. In *Proc. of the UIST*, Santa Fe, NM.

Michelle X. Zhou and Min Chen. 2003. Automated generation of graphic sketches by example. In *IJCAI*, Acapulco, Mexico.

Michelle X. Zhou and Shimei Pan. 2001. Automated authoring of coherent multimedia discourse in conversation systems. In *ACM Multimedia*, Ottawa, Canada.

Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop

Nizar Habash and **Owen Rambow**
Center for Computational Learning Systems
Columbia University
New York, NY 10115, USA
{habash,rambow}@cs.columbia.edu

Abstract

We present an approach to using a morphological analyzer for tokenizing and morphologically tagging (including part-of-speech tagging) Arabic words in one process. We learn classifiers for individual morphological features, as well as ways of using these classifiers to choose among entries from the output of the analyzer. We obtain accuracy rates on all tasks in the high nineties.

1 Introduction

Arabic is a morphologically complex language.[1] The morphological analysis of a word consists of determining the values of a large number of (orthogonal) features, such as basic part-of-speech (i.e., noun, verb, and so on), voice, gender, number, information about the clitics, and so on.[2] For Arabic, this gives us about 333,000 theoretically possible completely specified morphological analyses, i.e., morphological tags, of which about 2,200 are actually used in the first 280,000 words of the Penn Arabic Treebank (ATB). In contrast, English morphological tagsets usually have about 50 tags, which cover all morphological variation.

As a consequence, **morphological disambiguation** of a word in context, i.e., choosing a complete morphological tag, cannot be done successfully using methods developed for English because of data sparseness. Hajič (2000) demonstrates convincingly that morphological disambiguation can be aided by a morphological analyzer, which, given a word without any context, gives us the set of all possible morphological tags. The only work on Arabic tagging that uses a corpus for training and evaluation (that we are aware of), (Diab et al., 2004), does not use a morphological analyzer. In this paper, we show that the use of a morphological analyzer outperforms other tagging methods for Arabic; to our knowledge, we present the best-performing wide-coverage tokenizer on naturally occurring input and the best-performing morphological tagger for Arabic.

2 General Approach

Arabic words are often ambiguous in their morphological analysis. This is due to Arabic's rich system of affixation and clitics and the omission of disambiguating short vowels and other orthographic diacritics in standard orthography ("undiacritized orthography"). On average, a word form in the ATB has about 2 morphological analyses. An example of a word with some of its possible analyses is shown in Figure 1. Analyses 1 and 4 are both nouns. They differ in that the first noun has no affixes, while the second noun has a conjunction prefix (+و +*w* 'and') and a pronominal possessive suffix (ي+ +*y* 'my').

In our approach, tokenizing and morphologically tagging (including part-of-speech tagging) are the same operation, which consists of three phases. First, we obtain from our morphological analyzer a list of all possible analyses for the words of a given sentence. We discuss the data and our lexicon in

[1]We would like to thank Mona Diab for helpful discussions. The work reported in this paper was supported by NSF Award 0329163. The authors are listed in alphabetical order.

[2]In this paper, we only discuss inflectional morphology. Thus, the fact that the stem is composed of a root, a pattern, and an infi x vocalism is not relevant except as it affects broken plurals and verb aspect.

#	lexeme	gloss	POS	Conj	Part	Pron	Det	Gen	Num	Per	Voice	Asp
1	wAliy	ruler	N	NO	NO	NO	NO	masc	sg	3	NA	NA
2	<ilaY	and to me	P	YES	NO	YES	NA	NA	NA	NA	NA	NA
3	waliy	and I follow	V	YES	NO	NO	NA	neut	sg	1	act	imp
4	|l	and my clan	N	YES	NO	YES	NO	masc	sg	3	NA	NA
5	|liy~	and automatic	AJ	YES	NO	NO	NO	masc	sg	3	NA	NA

Figure 1: Possible analyses for the word والي *wAly*

more detail in Section 4.

Second, we apply classifiers for ten morphological features to the words of the text. The full list of features is shown in Figure 2, which also identifies possible values and which word classes (POS) can express these features. We discuss the training and decoding of these classifiers in Section 5.

Third, we choose among the analyses returned by the morphological analyzer by using the output of the classifiers. This is a non-trivial task, as the classifiers may not fully disambiguate the options, or they may be contradictory, with none of them fully matching any one choice. We investigate different ways of making this choice in Section 6.

As a result of this process, we have the original text, with each word augmented with values for all the features in Figure 2. These values represent a complete morphological disambiguation. Furthermore, these features contain enough information about the presence of clitics and affixes to perform tokenization, for any reasonable tokenization scheme. Finally, we can determine the POS tag, for any morphologically motivated POS tagset. Thus, we have performed tokenization, traditional POS tagging, and full morphological disambiguation in one fell swoop.

3 Related Work

Our work is inspired by Hajič (2000), who convincingly shows that for five Eastern European languages with complex inflection plus English, using a morphological analyzer[3] improves performance of a tagger. He concludes that for highly inflectional languages "the use of an independent morphological dictionary is the preferred choice [over] more annotated data". Hajič (2000) uses a general exponential model to predict each morphological feature separately (such as the ones we have listed in Figure 2), but he trains different models for each ambiguity left unresolved by the morphological analyzer, rather than training general models. For all languages, the use of a morphological analyzer results in tagging error reductions of at least 50%.

We depart from Hajič's work in several respects. First, we work on Arabic. Second, we use this approach to also perform tokenization. Third, we use the SVM-based Yamcha (which uses Viterbi decoding) rather than an exponential model; however, we do not consider this difference crucial and do not contrast our learner with others in this paper. Fourth, and perhaps most importantly, we do not use the notion of ambiguity class in the feature classifiers; instead we investigate different ways of using the results of the individual feature classifiers in directly choosing among the options produced for the word by the morphological analyzer.

While there have been many publications on computational morphological analysis for Arabic (see (Al-Sughaiyer and Al-Kharashi, 2004) for an excellent overview), to our knowledge only Diab et al. (2004) perform a large-scale corpus-based evaluation of their approach. They use the same SVM-based learner we do, Yamcha, for three different tagging tasks: word tokenization (tagging on letters of a word), which we contrast with our work in Section 7; POS tagging, which we discuss in relation to our work in Section 8; and base phrase chunking, which we do not discuss in this paper. We take the comparison between our results on POS tagging and those of Diab et al. (2004) to indicate that the use of a morphological analyzer is beneficial for Arabic as

[3]Hajič uses a lookup table, which he calls a "dictionary". The distinction between table-lookup and actual processing at run-time is irrelevant for us.

Feature Name	Description	Possible Values	POS that Carry Feature	Default
POS	Basic part-of-speech	See Footnote 9	all	X
Conj	Is there a cliticized conjunction?	YES, NO	all	NO
Part	Is there a cliticized particle?	YES, NO	all	NO
Pron	Is there a pronominal clitic?	YES, NO	V, N, PN, AJ, P, Q	NO
Det	Is there a cliticized definite determiner +ال *Al*+?	YES, NO	N, PN, AJ	NO
Gen	Gender (intrinsic or by agreement)	masc(uline), fem(inine), neut(er)	V, N, PN, AJ, PRO, REL, D	masc
Num	Number	sg (singular), du(al), pl(ural)	V, N, PN, AJ, PRO, REL, D	sg
Per	Person	1, 2, 3	V, N, PN, PRO	3
Voice	Voice	act(ive), pass(ive)	V	act
Asp	Aspect	imp(erfective), perf(ective), imperative	V	perf

Figure 2: Complete list of morphological features expressed by Arabic morphemes that we tag; the last column shows on which parts-of-speech this feature can be expressed; the value 'NA' is used for each feature other than POS, Conj, and Part if the word is not of the appropriate POS

well.

Several other publications deal specifically with segmentation. Lee et al. (2003) use a corpus of manually segmented words, which appears to be a subset of the first release of the ATB (110,000 words), and thus comparable to our training corpus. They obtain a list of prefixes and suffixes from this corpus, which is apparently augmented by a manually derived list of other affixes. Unfortunately, the full segmentation criteria are not given. Then a trigram model is learned from the segmented training corpus, and this is used to choose among competing segmentations for words in running text. In addition, a huge unannotated corpus (155 million words) is used to iteratively learn additional stems. Lee et al. (2003) show that the unsupervised use of the large corpus for stem identification increases accuracy. Overall, their error rates are higher than ours (2.9% vs. 0.7%), presumably because they do not use a morphological analyzer.

There has been a fair amount of work on entirely unsupervised segmentation. Among this literature, Rogati et al. (2003) investigate unsupervised learning of stemming (a variant of tokenization in which only the stem is retained) using Arabic as the example language. Unsurprisingly, the results are much worse than in our resource-rich approach. Darwish (2003) discusses unsupervised identification of roots; as mentioned above, we leave root identification to future work.

4 Preparing the Data

The data we use comes from the Penn Arabic Treebank (Maamouri et al., 2004). Like the English Penn Treebank, the corpus is a collection of news texts. Unlike the English Penn Treebank, the ATB is an ongoing effort, which is being released incrementally. As can be expected in this situation, the annotation has changed in subtle ways between the incremental releases. Even within one release (especially the first) there can be inconsistencies in the annotation. As our approach builds on linguistic knowledge, we need to carefully study how linguistic facts are represented in the ATB. In this section, we briefly summarize how we obtained the data in the representation we use for our machine learning experiments.[4]

We use the first two releases of the ATB, ATB1 and ATB2, which are drawn from different news sources. We divided both ATB1 and ATB2 into de-

[4]The code used to obtain the representations is available from the authors upon request.

velopment, training, and test corpora with roughly 12,000 word tokens in each of the development and test corpora, and 120,000 words in each of the training corpora. We will refer to the training corpora as TR1 and TR2, and to the test corpora as, TE1 and TE2. We report results on both TE1 and TE2 because of the differences in the two parts of the ATB, both in terms of origin and in terms of data preparation.

We use the ALMORGEANA morphological analyzer (Habash, 2005), a lexeme-based morphological generator and analyzer for Arabic.[5] A sample output of the morphological analyzer is shown in Figure 1. ALMORGEANA uses the databases (i.e., lexicon) from the Buckwalter Arabic Morphological Analyzer, but (in analysis mode) produces an output in the lexeme-and-feature format (which we need for our approach) rather than the stem-and-affix format of the Buckwalter analyzer. We use the data from first version of the Buckwalter analyzer (Buckwalter, 2002). The first version is fully consistent with neither ATB1 nor ATB2.

Our training data consists of a set of all possible morphological analyses for each word, with the unique correct analysis marked. Since we want to learn to choose the correct output using the features generated by ALMORGEANA, the training data must also be in the ALMORGEANA output format. To obtain this data, we needed to match data in the ATB to the lexeme-and-feature representation output by ALMORGEANA. The matching included the use of some heuristics, since the representations and choices are not always consistent in the ATB. For example, نحو *nHw* ‘towards’ is tagged as AV, N, or V (in the same syntactic contexts). We verified whether we introduced new errors while creating our data representation by manually inspecting 400 words chosen at random from TR1 and TR2. In eight cases, our POS tag differed from that in the ATB file; all but one case were plausible changes among Noun, Adjective, Adverb and Proper Noun resulting from missing entries in the Buckwalter’s lexicon. The remaining case was a failure in the conversion process relating to the handling of broken plurals at the lexeme level. We conclude that our data representation provides an adequate basis for performing machine learning experiments.

[5]The ALMORGEANA engine is available at http://clipdemos.umiacs.umd.edu/ALMORGEANA/.

An important issue in using morphological analyzers for morphological disambiguation is what happens to *unanalyzed* words, i.e., words that receive no analysis from the morphological analyzer. These are frequently proper nouns; a typical example is برلوسكوني *brlwskwny* ‘Berlusconi’, for which no entry exists in the Buckwalter lexicon. A backoff analysis mode in ALMORGEANA uses the morphological databases of prefixes, suffixes, and allowable combinations from the Buckwalter analyzer to hypothesize all possible stems along with feature sets. Our Berlusconi example yields 41 possible analyses, including the correct one (as a singular masculine PN). Thus, with the backoff analysis, unanalyzed words are distinguished for us only by the larger number of possible analyses (making it harder to choose the correct analysis). There are not many unanalyzed words in our corpus. In TR1, there are only 22 such words, presumably because the Buckwalter lexicon our morphological analyzer uses was developed onTR1. In TR2, we have 737 words without analysis (0.61% of the entire corpus, giving us a coverage of about 99.4% on domain-similar text for the Buckwalter lexicon).

In ATB1, and to a lesser degree in ATB2, some words have been given no morphological analysis. (These cases are not necessarily the same words that our morphological analyzer cannot analyze.) The POS tag assigned to these words is then NO_FUNC. In TR1 (138,756 words), we have 3,088 NO_FUNC POS labels (2.2%). In TR2 (168,296 words), the number of NO_FUNC labels has been reduced to 853 (0.5%). Since for these cases, there is no meaningful solution in the data, we have removed them from the evaluation (but not from training). In contrast, Diab et al. (2004) treat NO_FUNC like any other POS tag, but it is unclear whether this is meaningful. Thus, when comparing results from different approaches which make different choices about the data (for example, the NO_FUNC cases), one should bear in mind that small differences in performance are probably not meaningful.

5 Classifiers for Linguistic Features

We now describe how we train classifiers for the morphological features in Figure 2. We train one classifier per feature. We use Yamcha (Kudo and Matsumoto, 2003), an implementation of support vector machines which includes Viterbi decoding.[6] As training features, we use two sets. These sets are based on the ten morphological features in Figure 2, plus four other "hidden" morphological features, for which we do not train classifiers, but which are represented in the analyses returned by the morphological analyzer. The reason we do not train classifiers for the hidden features is that they are only returned by the morphological analyzer when they are marked overtly in orthography, but they are not disambiguated in case they are not overtly marked. The features are indefiniteness (presence of nunation), idafa (possessed), case, and mood. First, for each of the 14 morphological features and for each possible value (including 'NA' if applicable), we define a binary machine learning feature which states whether in *any* morphological analysis for that word, the feature has that value. This gives us 58 machine learning features per word. In addition, we define a second set of features which abstracts over the first set: for all features, we state whether any morphological analysis for that word has a value other than 'NA'. This yields a further 11 machine learning features (as 3 morphological features never have the value 'NA'). In addition, we use the untokenized word form and a binary feature stating whether there is an analysis or not. This gives us a total of 71 machine learning features per word. We specify a window of two words preceding and following the current word, using all 71 features for each word in this 5-word window. In addition, two dynamic features are used, namely the classification made for the preceding two words. For each of the ten classifiers, Yamcha then returns a confidence value for each possible value of the classifier, and in addition it marks the value that is chosen during subsequent Viterbi decoding (which need not be the value with the highest confidence value because of the inclusion of dynamic features).

We train on TR1 and report the results for the ten Yamcha classifiers on TE1 and TE2, using all simple tokens,[7] including punctuation, in Figure 3. The baseline BL is the most common value associated in the training corpus TR1 with every feature for a given word form (unigram). We see that the baseline for TE1 is quite high, which we assume is due to the fact that when there is ambiguity, often one interpretation is much more prevelant than the others. The error rates on the baseline approximately double on TE2, reflecting the difference between TE2 and TR1, and the small size of TR1. The performance of our classifiers is good on TE1 (third column), and only slightly worse on TE2 (fifth column). We attribute the increase in error reduction over the baseline for TE2 to successfully learned generalizations.

Method	BL	Class	BL	Class
Test	TE1	TE1	TE2	TE2
POS	96.6	97.7	91.1	95.5
Conj	99.9	99.9	99.7	99.9
Part	99.9	99.9	99.5	99.7
Pron	99.5	99.6	98.8	99.0
Det	98.8	99.2	96.8	98.3
Gen	98.6	99.2	95.8	98.2
Num	98.8	99.4	96.8	98.8
Per	97.6	98.7	94.8	98.1
Voice	98.8	99.3	97.5	99.0
Asp	98.8	99.4	97.4	99.1

Figure 3: Accuracy of classifiers (Class) for morphological features trained on TR1, and evaluated on TE1 and TE2; BL is the unigram baseline trained on TR1

We investigated the performance of the classifiers on unanalyzed words. The performance is generally below the baseline BL. We attribute this to the almost complete absence of unanalyzed words in training data TR1. In future work we could attempt to improve performance in these cases; however, given their small number, this does not seem a priority.

[6] We use Yamcha's default settings: standard SVM with 2nd degree polynomial kernel and 1 slack variable.

[7] We use the term *orthographic token* to designate tokens determined only by white space, while *simple tokens* are orthographic tokens from which punctuation has been segmented (becoming its own token), and from which all tatweels (the elongation character) have been removed.

6 Choosing an Analysis

Once we have the results from the classifiers for the ten morphological features, we combine them to choose an analysis from among those returned by the morphological analyzer. We investigate several options for how to do this combination. In the following, we use two numbers for each analysis. First, the **agreement** is the number of classifiers agreeing with the analysis. Second, the **weighted agreement** is the sum, over all classifiers, of the classification confidence measure of that value that agrees with the analysis. The agreement, but not the weighted agreement, uses Yamcha's Viterbi decoding.

• The majority combiner (Maj) chooses the analysis with the largest agreement.

• The confidence-based combiner (Con) chooses the analysis with the largest weighted agreement.

• The additive combiner (Add) chooses the analysis with the largest sum of agreement and weighted agreement.

• The multiplicative combiner (Mul) chooses the analysis with the largest product of agreement and weighted agreement.

• We use Ripper (Cohen, 1996) to learn a rule-based classifier (Rip) to determine whether an analysis from the morphological analyzer is a "good" or a "bad" analysis. We use the following features for training: for each morphological feature in Figure 2, we state whether or not the value chosen by its classifier agrees with the analysis, and with what confidence level. In addition, we use the word form. (The reason we use Ripper here is because it allows us to learn lower bounds for the confidence score features, which are real-valued.) In training, only the correct analysis is good. If exactly one analysis is classified as good, we choose that, otherwise we use Maj to choose.

• The baseline (BL) chooses the analysis most commonly assigned in TR1 to the word in question. For unseen words, the choice is made randomly.

In all cases, any remaining ties are resolved randomly.

We present the performance in Figure 4. We see that the best performing combination algorithm on TE1 is Maj, and on TE2 it is Rip. Recall that the Yamcha classifiers are trained on TR1; in addition, Rip is trained on the output of these Yamcha classifiers on TR2. The difference in performance between TE1 and TE2 shows the difference between the ATB1 and ATB2 (different source of news, and also small differences in annotation). However, the results for Rip show that retraining the Rip classifier on a new corpus can improve the results, without the need for retraining all ten Yamcha classifiers (which takes considerable time).

Corpus	TE1		TE2	
Method	All	Words	All	Words
BL	92.1	90.2	87.3	85.3
Maj	96.6	95.8	94.1	93.2
Con	89.9	87.6	88.9	87.2
Add	91.6	89.7	90.7	89.2
Mul	96.5	95.6	94.3	93.4
Rip	96.2	95.3	94.8	94.0

Figure 4: Results (percent accuracy) on choosing the correct analysis, measured per token (including and excluding punctuation and numbers); BL is the baseline

Figure 4 presents the accuracy of tagging using the whole complex morphological tagset. We can project this complex tagset to a simpler tagset, for example, POS. Then the minimum tagging accuracy for the simpler tagset must be greater than or equal to the accuracy of the complex morphological tagset. Even if a combining algorithm chooses the wrong analysis (and this is counted as a failure for the evaluation in this section), the chosen analysis may agree with some of the correct morphological features. We discuss our performance on the POS feature in Section 8.

7 Evaluating Tokenization

The term "tokenization" refers to the segmenting of a naturally occurring input sequence of orthographic symbols into elementary symbols ("tokens") used in subsequent processing steps (such as parsing) as basic units. In our approach, we determine all morphological properties of a word at once, so we can use this information to determine tokenization. There is not a single possible or obvious tokenization scheme: a tokenization scheme is an analytical tool devised by the researcher. We evaluate in this section how well our morphological disambiguation

Meth.	Word Acc.	Token Acc.	Token Prec.	Token Rec.	Token F-m.
BL	99.1	99.6	98.6	99.1	98.8
Maj	99.3	99.6	98.9	99.3	99.1

Figure 5: Results of tokenization on TE1: word accuracy measures for each input word whether it gets tokenized correctly, independently of the number of resulting tokens; the token-based measures refer to the four token fields into which the ATB splits each word

determines the ATB tokenization. The ATB starts with a simple tokenization, and then splits the word into four fields: conjunctions; particles (prepositions in the case of nouns); the word stem; and pronouns (object clitics in the case of verbs, possessive clitics in the case of nouns). The ATB does not tokenize the definite article +ال *Al+*.

We compare our output to the morphologically analyzed form of the ATB, and determine if our morphological choices lead to the correct identification of those clitics that need to be stripped off.[8] For our evaluation, we only choose the Maj chooser, as it performed best on TE1. We evaluate in two ways. In the first evaluation, we determine for each simple input word whether the tokenization is correct (no matter how many ATB tokens result). We report the percentage of words which are correctly tokenized in the second column in Figure 5. In the second evaluation, we report on the number of output tokens. Each word is divided into exactly four token fields, which can be either filled or empty (in the case of the three clitic token fields) or correct or incorrect (in the case of the stem token field). We report in Figure 5 accuracy over all token fields for all words in the test corpus, as well as recall, precision, and f-measure for the non-null token fields. The baseline BL is the tokenization associated with the morphological analysis most frequently chosen for the input word in training.

[8]The ATB generates normalized forms of certain clitics and of the word stem, so that the resulting tokens are not simply the result of splitting the original words. We do not actually generate the surface token form from our deep representation, but this can be done in a deterministic, rule-based manner, given our rich morphological analysis, e.g., by using ALMORGEANA in generation mode after splitting off all separable tokens.

While the token-based evaluation is identical to that performed by Diab et al. (2004), the results are not directly comparable as they did not use actual input words, but rather recreated input words from the regenerated tokens in the ATB. Sometimes this can simplify the analysis: for example, a ة *p (ta marbuta)* must be word-final in Arabic orthography, and thus a word-medial ة *p* in a recreated input word reliably signals a token boundary. The rather high baseline shows that tokenization is not a hard problem.

8 Evaluating POS Tagging

The POS tagset Diab et al. (2004) use is a subset of the tagset for English that was introduced with the English Penn Treebank. The large set of Arabic tags has been mapped (by the Linguistic Data Consortium) to this smaller English set, and the meaning of the English tags has changed. We consider this tagset unmotivated, as it makes morphological distinctions because they are marked in English, not Arabic. The morphological distinctions that the English tagset captures represent the complete morphological variation that can be found in English. However, in Arabic, much morphological variation goes untagged. For example, verbal inflections for subject person, number, and gender are not marked; dual and plural are not distinguished on nouns; and gender is not marked on nouns at all. In Arabic nouns, arguably the gender feature is the more interesting distinction (rather than the number feature) as verbs in Arabic always agree with their nominal subjects in gender. Agreement in number occurs only when the nominal subject precedes the verb. We use the tagset here only to compare to previous work. Instead, we advocate using a reduced part-of-speech tag set,[9] along with the other orthogonal linguistic features in Figure 2.

We map our best solutions as chosen by the Maj model in Section 6 to the English tagset, and we furthermore assume (as do Diab et al. (2004)) the gold standard tokenization. We then evaluate against the gold standard POS tagging which we have mapped

[9] We use V (Verb), N (Noun), PN (Proper Noun), AJ (Adjective), AV (Adverb), PRO (Nominal Pronoun), P (Preposition/Particle), D (Determiner), C (Conjunction), NEG (Negative particle), NUM (Number), AB (Abbreviation), IJ (Interjection), PX (Punctuation), and X (Unknown).

Corpus		TE1		TE2	
Method	Tags	All	Words	All	Words
BL	PTB	93.9	93.3	90.9	89.8
	Smp	94.9	94.3	92.6	91.4
Maj	PTB	97.6	97.5	95.7	95.2
	Smp	98.1	97.8	96.5	96.0

Figure 6: Part-of-speech tagging accuracy measured for all tokens (based on gold-standard tokenization) and only for word tokens, using the Penn Treebank (PTB) tagset as well as the smaller tagset (Smp) (see Footnote 9); BL is the baseline obtained by using the POS value from the baseline tag used in Section 6

similarly. We obtain a score for TE1 of 97.6% on all tokens. Diab et al. (2004) report a score of 95.5% for all tokens on a test corpus drawn from ATB1, thus their figure is comparable to our score of 97.6%. On our own reduced POS tagset, evaluating on TE1, we obtain an accuracy score of 98.1% on all tokens. The full dataset is shown in Figure 6.

9 Conclusion and Outlook

We have shown how to use a morphological analyzer for tokenization, part-of-speech tagging, and morphological disambiguation in Arabic. We have shown that the use of a morphological analyzer is beneficial in POS tagging, and we believe our results are the best published to date for tokenization of naturally occurring input (in undiacritized orthography) and POS tagging.

We intend to apply our approach to Arabic dialects, for which currently no annotated corpora exist, and for which very few written corpora of any kind exist (making the dialects bad candidates even for unsupervised learning). However, there is a fair amount of descriptive work on dialectal morphology, so that dialectal morphological analyzers may be easier to come by than dialect corpora. We intend to explore to what extent we can transfer models trained on Standard Arabic to dialectal morphological disambiguation.

References

Imad A. Al-Sughaiyer and Ibrahim A. Al-Kharashi. 2004. Arabic morphological analysis techniques: A comprehensive survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213.

Tim Buckwalter. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49.

William Cohen. 1996. Learning trees and rules with set-valued features. In *Fourteenth Conference of the American Association of Artificial Intelligence*. AAAI.

Kareem Darwish. 2003. Building a shallow Arabic morphological analyser in one day. In *ACL02 Workshop on Computational Approaches to Semitic Languages*, Philadelpia, PA. Association for Computational Linguistics.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of arabic text: From raw text to base phrase chunks. In *5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04)*, Boston, MA.

Nizar Habash. 2005. Arabic morphological representations for machine translation. In Abdelhadi Soudi, Antal van den Bosch, and Guenter Neumann, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, Text, Speech, and Language Technology. Kluwer/Springer. in press.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'00)*, Seattle, WA.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *41st Meeting of the Association for Computational Linguistics (ACL'03)*, Sapporo, Japan.

Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. 2003. Language model based Arabic word segmentation. In *41st Meeting of the Association for Computational Linguistics (ACL'03)*, pages 399–406, Sapporo, Japan.

Mohamed Maamouri, Ann Bies, and Tim Buckwalter. 2004. The penn arabic treebank : Building a large-scale annotated arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Monica Rogati, J. Scott McCarley, and Yiming Yang. 2003. Unsupervised learning of arabic stemming using a parallel corpus. In *41st Meeting of the Association for Computational Linguistics (ACL'03)*, pages 391–398, Sapporo, Japan.

Semantic Role Labeling Using Different Syntactic Views[*]

Sameer Pradhan, Wayne Ward,
Kadri Hacioglu, James H. Martin
Center for Spoken Language Research,
University of Colorado,
Boulder, CO 80303
{spradhan,whw,hacioglu,martin}@cslr.colorado.edu

Daniel Jurafsky
Department of Linguistics,
Stanford University,
Stanford, CA 94305
jurafsky@stanford.edu

Abstract

Semantic role labeling is the process of annotating the predicate-argument structure in text with semantic labels. In this paper we present a state-of-the-art baseline semantic role labeling system based on Support Vector Machine classifiers. We show improvements on this system by: i) adding new features including features extracted from dependency parses, ii) performing feature selection and calibration and iii) combining parses obtained from semantic parsers trained using different syntactic views. Error analysis of the baseline system showed that approximately half of the argument identification errors resulted from parse errors in which there was no syntactic constituent that aligned with the correct argument. In order to address this problem, we combined semantic parses from a Minipar syntactic parse and from a chunked syntactic representation with our original baseline system which was based on Charniak parses. All of the reported techniques resulted in performance improvements.

1 Introduction

Semantic Role Labeling is the process of annotating the predicate-argument structure in text with semantic labels (Gildea and Jurafsky, 2000; Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Surdeanu et al., 2003; Hacioglu and Ward, 2003; Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Pradhan et al., 2004; Hacioglu, 2004). The architecture underlying all of these systems introduces two distinct sub-problems: the identification of syntactic constituents that are semantic roles for a given predicate, and the labeling of the those constituents with the correct semantic role.

A detailed error analysis of our baseline system indicates that the identification problem poses a significant bottleneck to improving overall system performance. The baseline system's accuracy on the task of labeling nodes known to represent semantic arguments is 90%. On the other hand, the system's performance on the identification task is quite a bit lower, achieving only 80% recall with 86% precision. There are two sources of these identification errors: i) failures by the system to identify all and only those constituents that correspond to semantic roles, *when those constituents are present in the syntactic analysis*, and ii) failures by the syntactic analyzer to provide the constituents that align with correct arguments. The work we present here is tailored to address these two sources of error in the identification problem.

The remainder of this paper is organized as follows. We first describe a baseline system based on the best published techniques. We then report on two sets of experiments using techniques that improve performance on the problem of finding arguments when they are present in the syntactic analysis. In the first set of experiments we explore new

[*] This research was partially supported by the ARDA AQUAINT program via contract OCG4423B and by the NSF via grants IS-9978025 and ITR/HCI 0086132

features, including features extracted from a parser that provides a different syntactic view – a Combinatory Categorial Grammar (CCG) parser (Hockenmaier and Steedman, 2002). In the second set of experiments, we explore approaches to identify optimal subsets of features for each argument class, and to calibrate the classifier probabilities.

We then report on experiments that address the problem of arguments missing from a given syntactic analysis. We investigate ways to combine hypotheses generated from semantic role taggers trained using different syntactic views – one trained using the Charniak parser (Charniak, 2000), another on a rule-based dependency parser – Minipar (Lin, 1998), and a third based on a flat, shallow syntactic chunk representation (Hacioglu, 2004a). We show that these three views complement each other to improve performance.

2 Baseline System

For our experiments, we use Feb 2004 release of PropBank[1] (Kingsbury and Palmer, 2002; Palmer et al., 2005), a corpus in which predicate argument relations are marked for verbs in the Wall Street Journal (WSJ) part of the Penn TreeBank (Marcus et al., 1994). PropBank was constructed by assigning semantic arguments to constituents of hand-corrected TreeBank parses. Arguments of a verb are labeled ARG0 to ARG5, where ARG0 is the PROTO-AGENT, ARG1 is the PROTO-PATIENT, etc. In addition to these CORE ARGUMENTS, additional ADJUNCTIVE ARGUMENTS, referred to as ARGMs are also marked. Some examples are ARGM-LOC, for locatives; ARGM-TMP, for temporals; ARGM-MNR, for manner, etc. Figure 1 shows a syntax tree along with the argument labels for an example extracted from PropBank. We use Sections 02-21 for training, Section 00 for development and Section 23 for testing.

We formulate the semantic labeling problem as a multi-class classification problem using Support Vector Machine (SVM) classifier (Hacioglu et al., 2003; Pradhan et al., 2003; Pradhan et al., 2004) TinySVM[2] along with YamCha[3] (Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2001) are used to implement the system. Using what is known as the ONE VS ALL classification strategy, n binary classifiers are trained, where n is number of semantic classes including a NULL class.

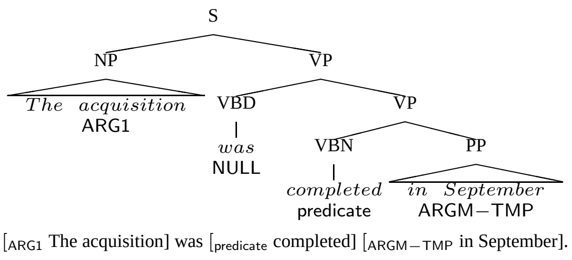


Figure 1: Syntax tree for a sentence illustrating the PropBank tags.

The baseline feature set is a combination of features introduced by Gildea and Jurafsky (2002) and ones proposed in Pradhan et al., (2004), Surdeanu et al., (2003) and the *syntactic-frame* feature proposed in (Xue and Palmer, 2004). Table 1 lists the features used.

PREDICATE LEMMA
PATH: Path from the constituent to the predicate in the parse tree.
POSITION: Whether the constituent is before or after the predicate.
VOICE
PREDICATE SUB-CATEGORIZATION
PREDICATE CLUSTER
HEAD WORD: Head word of the constituent.
HEAD WORD POS: POS of the head word
NAMED ENTITIES IN CONSTITUENTS: 7 named entities as 7 binary features.
PARTIAL PATH: Path from the constituent to the lowest common ancestor of the predicate and the constituent.
VERB SENSE INFORMATION: Oracle verb sense information from PropBank
HEAD WORD OF PP: Head of PP replaced by head word of NP inside it, and PP replaced by PP-*preposition*
FIRST AND LAST WORD/POS IN CONSTITUENT
ORDINAL CONSTITUENT POSITION
CONSTITUENT TREE DISTANCE
CONSTITUENT RELATIVE FEATURES: Nine features representing the phrase type, head word and head word part of speech of the parent, and left and right siblings of the constituent.
TEMPORAL CUE WORDS
DYNAMIC CLASS CONTEXT
SYNTACTIC FRAME
CONTENT WORD FEATURES: Content word, its POS and named entities in the content word

Table 1: Features used in the Baseline system

As described in (Pradhan et al., 2004), we post-process the n-best hypotheses using a trigram language model of the argument sequence.

We analyze the performance on three tasks:

- **Argument Identification** – This is the process of identifying the parsed constituents in the sentence that represent semantic arguments of a given predicate.

[1]http://www.cis.upenn.edu/˜ace/
[2]http://chasen.org/˜taku/software/TinySVM/
[3]http://chasen.org/˜taku/software/yamcha/

- **Argument Classification** – Given constituents known to represent arguments of a predicate, assign the appropriate argument labels to them.
- **Argument Identification and Classification** – A combination of the above two tasks.

ALL ARGS	Task	P (%)	R (%)	F_1	A (%)
HAND	Id.	96.2	95.8	96.0	
	Classification	-	-	-	93.0
	Id. + Classification	89.9	89.0	89.4	
AUTOMATIC	Id.	86.8	80.0	83.3	
	Classification	-	-	-	90.1
	Id. + Classification	80.9	76.8	78.8	

Table 2: Baseline system performance on all tasks using hand-corrected parses and automatic parses on PropBank data.

Table 2 shows the performance of the system using the hand corrected, TreeBank parses (HAND) and using parses produced by a Charniak parser (AUTOMATIC). Precision (P), Recall (R) and F_1 scores are given for the identification and combined tasks, and Classification Accuracy (A) for the classification task.

Classification performance using Charniak parses is about 3% absolute worse than when using TreeBank parses. On the other hand, argument identification performance using Charniak parses is about 12.7% absolute worse. Half of these errors – about 7% are due to missing constituents, and the other half – about 6% are due to mis-classifications.

Motivated by this severe degradation in argument identification performance for automatic parses, we examined a number of techniques for improving argument identification. We made a number of changes to the system which resulted in improved performance. The changes fell into three categories: i) new features, ii) feature selection and calibration, and iii) combining parses from different syntactic representations.

3 Additional Features

3.1 CCG Parse Features

While the Path feature has been identified to be very important for the argument identification task, it is one of the most sparse features and may be difficult to train or generalize (Pradhan et al., 2004; Xue and Palmer, 2004). A dependency grammar should generate shorter paths from the predicate to dependent words in the sentence, and could be a more robust complement to the phrase structure grammar paths extracted from the Charniak parse tree. Gildea and Hockenmaier (2003) report that using features extracted from a Combinatory Categorial Grammar (CCG) representation improves semantic labeling performance on core arguments. We evaluated features from a CCG parser combined with our baseline feature set. We used three features that were introduced by Gildea and Hockenmaier (2003):

- **Phrase type –** This is the category of the maximal projection between the two words – the predicate and the dependent word.
- **Categorial Path –** This is a feature formed by concatenating the following three values: i) category to which the dependent word belongs, ii) the direction of dependence and iii) the slot in the category filled by the dependent word.
- **Tree Path –** This is the categorial analogue of the path feature in the Charniak parse based system, which traces the path from the dependent word to the predicate through the binary CCG tree.

Parallel to the hand-corrected TreeBank parses, we also had access to correct CCG parses derived from the TreeBank (Hockenmaier and Steedman, 2002a). We performed two sets of experiments. One using the correct CCG parses, and the other using parses obtained using StatCCG[4] parser (Hockenmaier and Steedman, 2002). We incorporated these features in the systems based on hand-corrected TreeBank parses and Charniak parses respectively. For each constituent in the Charniak parse tree, if there was a dependency between the head word of the constituent and the predicate, then the corresponding CCG features for those words were added to the features for that constituent. Table 3 shows the performance of the system when these features were added. The corresponding baseline performances are mentioned in parentheses.

3.2 Other Features

We added several other features to the system. Position of the clause node (S, SBAR) seems to be

[4]Many thanks to Julia Hockenmaier for providing us with the CCG bank as well as the StatCCG parser.

ALL ARGS	Task	P (%)	R (%)	F_1
HAND	Id.	97.5 (96.2)	96.1 (95.8)	96.8 (96.0)
	Id. + Class.	91.8 (89.9)	90.5 (89.0)	91.2 (89.4)
AUTOMATIC	Id.	87.1 (86.8)	80.7 (80.0)	83.8 (83.3)
	Id. + Class.	81.5 (80.9)	77.2 (76.8)	79.3 (78.8)

Table 3: Performance improvement upon adding CCG features to the Baseline system.

an important feature in argument identification (Hacioglu et al., 2004) therefore we experimented with four clause-based path feature variations. We added the predicate context to capture predicate sense variations. For some adjunctive arguments, punctuation plays an important role, so we added some punctuation features. All the new features are shown in Table 4

CLAUSE-BASED PATH VARIATIONS: I. Replacing all the nodes in a path other than clause nodes with an "*". For example, the path NP↑S↑VP↑SBAR↑NP↑VP↓VBD becomes NP↑S↑*S↑*↑*↓VBD II. Retaining only the clause nodes in the path, which for the above example would produce NP↑S↑S↓VBD, III. Adding a binary feature that indicates whether the constituent is in the same clause as the predicate, IV. collapsing the nodes between S nodes which gives NP↑S↑NP↑VP↓VBD.
PATH N-GRAMS: This feature decomposes a path into a series of trigrams. For example, the path NP↑S↑VP↑SBAR↑NP↑VP↓VBD becomes: NP↑S↑VP, S↑VP↑SBAR, VP↑SBAR↑NP, SBAR↑NP↑VP, etc. We used the first ten trigrams as ten features. Shorter paths were padded with nulls.
SINGLE CHARACTER PHRASE TAGS: Each phrase category is clustered to a category defined by the first character of the phrase label.
PREDICATE CONTEXT: Two words and two word POS around the predicate and including the predicate were added as ten new features.
PUNCTUATION: Punctuation before and after the constituent were added as two new features.
FEATURE CONTEXT: Features for argument bearing constituents were added as features to the constituent being classified.

Table 4: Other Features

4 Feature Selection and Calibration

In the baseline system, we used the same set of features for all the n binary ONE VS ALL classifiers. Error analysis showed that some features specifically suited for one argument class, for example, core arguments, tend to hurt performance on some adjunctive arguments. Therefore, we thought that selecting subsets of features for each argument class might improve performance. To achieve this, we performed a simple feature selection procedure. For each argument, we started with the set of features introduced by (Gildea and Jurafsky, 2002). We pruned this set by training classifiers after leaving out one feature at a time and checking its performance on a development set. We used the χ^2 significance while making pruning decisions. Following that, we added each of the other features one at a time to the pruned baseline set of features and selected ones that showed significantly improved performance. Since the feature selection experiments were computationally intensive, we performed them using 10k training examples.

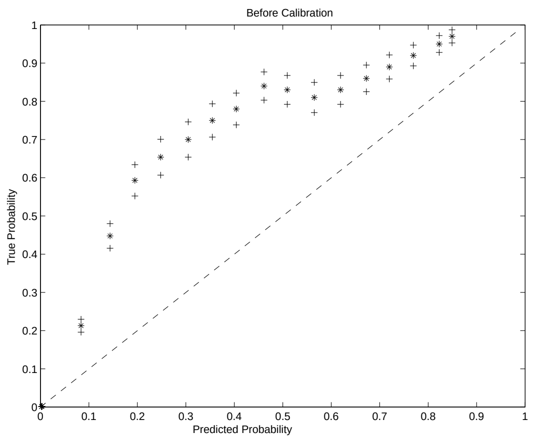
SVMs output distances not probabilities. These distances may not be comparable across classifiers, especially if different features are used to train each binary classifier. In the baseline system, we used the algorithm described by Platt (Platt, 2000) to convert the SVM scores into probabilities by fitting to a sigmoid. When all classifiers used the same set of features, fitting all scores to a single sigmoid was found to give the best performance. Since different feature sets are now used by the classifiers, we trained a separate sigmoid for each classifier.

	Raw Scores	Probabilities	
		After lattice-rescoring	
		Uncalibrated	Calibrated
	(%)	(%)	(%)
Same Feat. same sigmoid	74.7	74.7	75.4
Selected Feat. diff. sigmoids	75.4	75.1	76.2

Table 5: Performance improvement on selecting features per argument and calibrating the probabilities on 10k training data.

Foster and Stine (2004) show that the pool-adjacent-violators (PAV) algorithm (Barlow et al., 1972) provides a better method for converting raw classifier scores to probabilities when Platt's algorithm fails. The probabilities resulting from either conversions may not be properly calibrated. So, we binned the probabilities and trained a warping function to calibrate them. For each argument classifier, we used both the methods for converting raw SVM scores into probabilities and calibrated them using a development set. Then, we visually inspected the calibrated plots for each classifier and chose the method that showed better calibration as the calibration procedure for that classifier. Plots of the predicted probabilities versus true probabilities for the ARGM-TMP VS ALL classifier, before and after calibration are shown in Figure 2. The performance improvement over a classifier that is trained using all the features for all the classes is shown in Table 5.

Table 6 shows the performance of the system after adding the CCG features, additional features ex-



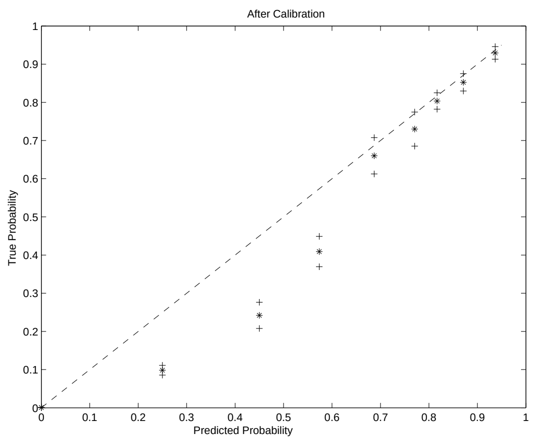


Figure 2: Plots showing true probabilities versus predicted probabilities before and after calibration on the test set for ARGM-TMP.

tracted from the Charniak parse tree, and performing feature selection and calibration. Numbers in parentheses are the corresponding baseline performances.

TASK	P (%)	R (%)	F_1	A (%)
Id.	86.9 (86.8)	84.2 (80.0)	85.5 (83.3)	
Class.	-	-	-	92.0 (90.1)
Id. + Class.	82.1 (80.9)	77.9 (76.8)	79.9 (78.8)	

Table 6: Best system performance on all tasks using automatically generated syntactic parses.

5 Alternative Syntactic Views

Adding new features can improve performance when the syntactic representation being used for classification contains the correct constituents. Additional features can't recover from the situation where the parse tree being used for classification doesn't contain the correct constituent representing an argument. Such parse errors account for about 7% absolute of the errors (or, about half of 12.7%) for the Charniak parse based system. To address these errors, we added two additional parse representations: i) Minipar dependency parser, and ii) chunking parser (Hacioglu et al., 2004). The hope is that these parsers will produce different errors than the Charniak parser since they represent different syntactic views. The Charniak parser is trained on the Penn TreeBank corpus. Minipar is a rule based dependency parser. The chunking parser is trained on PropBank and produces a flat syntactic representation that is very different from the full parse tree produced by Charniak. A combination of the three different parses could produce better results than any single one.

5.1 Minipar-based Semantic Labeler

Minipar (Lin, 1998; Lin and Pantel, 2001) is a rule-based dependency parser. It outputs dependencies between a word called *head* and another called *modifier*. Each word can modify at most one word. The dependency relationships form a dependency tree.

The set of words under each node in Minipar's dependency tree form a contiguous segment in the original sentence and correspond to the constituent in a constituent tree. We formulate the semantic labeling problem in the same way as in a constituent structure parse, except we classify the nodes that represent head words of constituents. A similar formulation using dependency trees derived from TreeBank was reported in Hacioglu (Hacioglu, 2004). In that experiment, the dependency trees were derived from hand-corrected TreeBank trees using head word rules. Here, an SVM is trained to assign PropBank argument labels to nodes in Minipar dependency trees using the following features:

Table 8 shows the performance of the Minipar-based semantic parser.

Minipar performance on the PropBank corpus is substantially worse than the Charniak based system. This is understandable from the fact that Minipar is not designed to produce constituents that would exactly match the constituent segmentation used in TreeBank. In the test set, about 37% of the argu-

PREDICATE LEMMA
HEAD WORD: The word representing the node in the dependency tree.
HEAD WORD POS: Part of speech of the head word.
POS PATH: This is the path from the predicate to the head word through the dependency tree connecting the part of speech of each node in the tree.
DEPENDENCY PATH: Each word that is connected to the head word has a particular dependency relationship to the word. These are represented as labels on the arc between the words. This feature is the dependencies along the path that connects two words.
VOICE
POSITION

Table 7: Features used in the Baseline system using Minipar parses.

Task	P	R	F$_1$
	(%)	(%)	
Id.	73.5	43.8	54.6
Id. + Classification	66.2	36.7	47.2

Table 8: Baseline system performance on all tasks using Minipar parses.

ments do not have corresponding constituents that match its boundaries. In experiments reported by Hacioglu (Hacioglu, 2004), a mismatch of about 8% was introduced in the transformation from hand-corrected constituent trees to dependency trees. Using an errorful automatically generated tree, a still higher mismatch would be expected. In case of the CCG parses, as reported by Gildea and Hockenmaier (2003), the mismatch was about 23%. A more realistic way to score the performance is to score tags assigned to head words of constituents, rather than considering the exact boundaries of the constituents as reported by Gildea and Hockenmaier (2003). The results for this system are shown in Table 9.

	Task	P	R	F$_1$
		(%)	(%)	
CHARNIAK	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
MINIPAR	Id.	83.3	61.1	70.5
	Id. + Classification	72.9	53.5	61.7

Table 9: Head-word based performance using Charniak and Minipar parses.

5.2 Chunk-based Semantic Labeler

Hacioglu has previously described a chunk based semantic labeling method (Hacioglu et al., 2004). This system uses SVM classifiers to first chunk input text into flat chunks or base phrases, each labeled with a syntactic tag. A second SVM is trained to assign semantic labels to the chunks. The system is trained on the PropBank training data.

WORDS
PREDICATE LEMMAS
PART OF SPEECH TAGS
BP POSITIONS: The position of a token in a BP using the IOB2 representation (e.g. B-NP, I-NP, O, etc.)
CLAUSE TAGS: The tags that mark token positions in a sentence with respect to clauses.
NAMED ENTITIES: The IOB tags of named entities.
TOKEN POSITION: The position of the phrase with respect to the predicate. It has three values as ”before”, ”after” and ”-” (for the predicate)
PATH: It defines a flat path between the token and the predicate
CLAUSE BRACKET PATTERNS
CLAUSE POSITION: A binary feature that identifies whether the token is inside or outside the clause containing the predicate
HEADWORD SUFFIXES: suffixes of headwords of length 2, 3 and 4.
DISTANCE: Distance of the token from the predicate as a number of base phrases, and the distance as the number of VP chunks.
LENGTH: the number of words in a token.
PREDICATE POS TAG: the part of speech category of the predicate
PREDICATE FREQUENCY: Frequent or rare using a threshold of 3.
PREDICATE BP CONTEXT: The chain of BPs centered at the predicate within a window of size -2/+2.
PREDICATE POS CONTEXT: POS tags of words immediately preceding and following the predicate.
PREDICATE ARGUMENT FRAMES: Left and right core argument patterns around the predicate.
NUMBER OF PREDICATES: This is the number of predicates in the sentence.

Table 10: Features used by chunk based classifier.

Table 10 lists the features used by this classifier. For each token (base phrase) to be tagged, a set of features is created from a fixed size context that surrounds each token. In addition to the above features, it also uses previous semantic tags that have already been assigned to the tokens contained in the linguistic context. A 5-token sliding window is used for the context.

	P	R	F_1
	(%)	(%)	
Id. and Classification	72.6	66.9	69.6

Table 11: Semantic chunker performance on the combined task of Id. and classification.

SVMs were trained for begin (B) and inside (I) classes of all arguments and outside (O) class for a total of 78 one-vs-all classifiers. Again, TinySVM[5] along with YamCha[6] (Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2001) are used as the SVM training and test software.

Table 11 presents the system performances on the PropBank test set for the chunk-based system.

[5]http://chasen.org/~taku/software/TinySVM/

[6]http://chasen.org/~taku/software/yamcha/

6 Combining Semantic Labelers

We combined the semantic parses as follows: i) scores for arguments were converted to calibrated probabilities, and arguments with scores below a threshold value were deleted. Separate thresholds were used for each parser. ii) For the remaining arguments, the more probable ones among overlapping ones were selected. In the chunked system, an argument could consist of a sequence of chunks. The probability assigned to the begin tag of an argument was used as the probability of the sequence of chunks forming an argument. Table 12 shows the performance improvement after the combination. Again, numbers in parentheses are respective baseline performances.

TASK	P	R	F$_1$
	(%)	(%)	
Id.	85.9 (86.8)	88.3 (80.0)	87.1 (83.3)
Id. + Class.	81.3 (80.9)	80.7 (76.8)	81.0 (78.8)

Table 12: Constituent-based best system performance on argument identification and argument identification and classification tasks after combining all three semantic parses.

The main contribution of combining both the Minipar based and the Charniak-based parsers was significantly improved performance on ARG1 in addition to slight improvements to some other arguments. Table 13 shows the effect on selected arguments on sentences that were altered during the the combination of Charniak-based and Chunk-based parses.

Number of Propositions	107
Percentage of perfect props before combination	0.00
Percentage of perfect props after combination	45.95

	Before			After		
	P	R	F$_1$	P	R	F$_1$
	(%)	(%)		(%)	(%)	
Overall	94.8	53.4	68.3	80.9	73.8	**77.2**
ARG0	96.0	85.7	90.5	92.5	89.2	**90.9**
ARG1	71.4	13.5	22.7	59.4	59.4	**59.4**
ARG2	100.0	20.0	33.3	50.0	20.0	28.5
ARGM-DIS	100.0	40.0	57.1	100.0	100.0	**100.0**

Table 13: Performance improvement on parses changed during pair-wise Charniak and Chunk combination.

A marked increase in number of propositions for which all the arguments were identified correctly from 0% to about 46% can be seen. Relatively few predicates, 107 out of 4500, were affected by this combination.

To give an idea of what the potential improvements of the combinations could be, we performed an oracle experiment for a combined system that tags head words instead of exact constituents as we did in case of Minipar-based and Charniak-based semantic parser earlier. In case of chunks, first word in prepositional base phrases was selected as the head word, and for all other chunks, the last word was selected to be the head word. If the correct argument was found present in either the Charniak, Minipar or Chunk hypotheses then that was selected. The results for this are shown in Table 14. It can be seen that the head word based performance almost approaches the constituent based performance reported on the hand-corrected parses in Table 3 and there seems to be considerable scope for improvement.

	Task	P	R	F$_1$
		(%)	(%)	
C	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
C+M	Id.	98.4	90.6	94.3
	Id. + Classification	93.1	86.0	89.4
C+CH	Id.	98.9	88.8	93.6
	Id. + Classification	92.5	83.3	87.7
C+M+CH	Id.	99.2	92.5	95.7
	Id. + Classification	94.6	88.4	91.5

Table 14: Performance improvement on head word based scoring after oracle combination. Charniak (C), Minipar (M) and Chunker (CH).

Table 15 shows the performance improvement in the actual system for pairwise combination of the parsers and one using all three.

	Task	P	R	F$_1$
		(%)	(%)	
C	Id.	92.2	87.5	89.8
	Id. + Classification	85.9	81.6	83.7
C+M	Id.	91.7	89.9	90.8
	Id. + Classification	85.0	83.9	84.5
C+CH	Id.	91.5	91.1	91.3
	Id. + Classification	84.9	84.3	84.7
C+M+CH	Id.	91.5	91.9	91.7
	Id. + Classification	85.1	85.5	85.2

Table 15: Performance improvement on head word based scoring after combination. Charniak (C), Minipar (M) and Chunker (CH).

7 Conclusions

We described a state-of-the-art baseline semantic role labeling system based on Support Vector Machine classifiers. Experiments were conducted to evaluate three types of improvements to the system: i) adding new features including features extracted from a Combinatory Categorial Grammar parse, ii) performing feature selection and calibration and iii) combining parses obtained from semantic parsers trained using different syntactic views. We combined semantic parses from a Minipar syntactic parse and from a chunked syntactic representation with our original baseline system which was based on Charniak parses. The belief was that semantic parses based on different syntactic views would make different errors and that the combination would be complimentary. A simple combination of these representations did lead to improved performance.

8 Acknowledgements

This research was partially supported by the ARDA AQUAINT program via contract OCG4423B and by the NSF via grants IS-9978025 and ITR/HCI 0086132. Computer time was provided by NSF ARI Grant #CDA-9601817, NSF MRI Grant #CNS-0420873, NASA AIST grant #NAG2-1646, DOE SciDAC grant #DE-FG02-04ER63870, NSF sponsorship of the National Center for Atmospheric Research, and a grant from the IBM Shared University Research (SUR) program.

We would like to thank Ralph Weischedel and Scott Miller of BBN Inc. for letting us use their named entity tagger – IdentiFinder; Martha Palmer for providing us with the PropBank data; Dan Gildea and Julia Hockenmaier for providing the gold standard CCG parser information, and all the anonymous reviewers for their helpful comments.

References

R. E. Barlow, D. J. Bartholomew, J. M. Bremmer, and H. D. Brunk. 1972. *Statistical Inference under Order Restrictions*. Wiley, New York.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139, Seattle, Washington.

John Chen and Owen Rambow. 2003. Use of deep linguistics features for the recognition and labeling of semantic arguments. In *Proceedings of the EMNLP*, Sapporo, Japan.

Dean P. Foster and Robert A. Stine. 2004. Variable selection in data mining: building a predictive model for bankruptcy. *Journal of American Statistical Association*, 99, pages 303–313.

Dan Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *Proceedings of the EMNLP*, Sapporo, Japan.

Daniel Gildea and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *Proceedings of ACL*, pages 512–520, Hong Kong, October.

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

Daniel Gildea and Martha Palmer. 2002. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of ACL*, Philadelphia, PA.

Kadri Hacioglu. 2004. Semantic role labeling using dependency trees. In *Proceedings of COLING*, Geneva, Switzerland.

Kadri Hacioglu and Wayne Ward. 2003. Target word detection and semantic role chunking using support vector machines. In *Proceedings of HLT/NAACL*, Edmonton, Canada.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James Martin, and Dan Jurafsky. 2003. Shallow semantic parsing using support vector machines. Technical Report TR-CSLR-2003-1, Center for Spoken Language Research, Boulder, Colorado.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of CoNLL-2004, Shared Task – Semantic Role Labeling*.

Kadri Hacioglu. 2004a. A lightweight semantic chunking model based on tagging. In *Proceedings of HLT/NAACL*, Boston, MA.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with combinatory grammars. In *Proceedings of the ACL*, pages 335–342.

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Canary Islands, Spain.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC*, Las Palmas, Canary Islands, Spain.

Taku Kudo and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of CoNLL and LLL*, pages 142–144.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the NAACL*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *In Workshop on the Evaluation of Parsing Systems*, Granada, Spain.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. To appear *Computational Linguistics*.

John Platt. 2000. Probabilities for support vector machines. In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT press, Cambridge, MA.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James Martin, and Dan Jurafsky. 2003. Semantic role parsing: Adding semantic structure to unstructured text. In *Proceedings of ICDM*, Melbourne, Florida.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL*, Boston, MA.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL*, Sapporo, Japan.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*, Barcelona, Spain.

Joint Learning Improves Semantic Role Labeling

Kristina Toutanova
Dept of Computer Science
Stanford University
Stanford, CA, 94305
kristina@cs.stanford.edu

Aria Haghighi
Dept of Computer Science
Stanford University
Stanford, CA, 94305
aria42@stanford.edu

Christopher D. Manning
Dept of Computer Science
Stanford University
Stanford, CA, 94305
manning@cs.stanford.edu

Abstract

Despite much recent progress on accurate semantic role labeling, previous work has largely used independent classifiers, possibly combined with separate label sequence models via Viterbi decoding. This stands in stark contrast to the linguistic observation that a core argument frame is a *joint* structure, with strong dependencies between arguments. We show how to build a joint model of argument frames, incorporating novel features that model these interactions into discriminative log-linear models. This system achieves an error reduction of 22% on all arguments and 32% on core arguments over a state-of-the art independent classifier for gold-standard parse trees on PropBank.

1 Introduction

The release of semantically annotated corpora such as FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2003) has made it possible to develop high-accuracy statistical models for automated semantic role labeling (Gildea and Jurafsky, 2002; Pradhan et al., 2004; Xue and Palmer, 2004). Such systems have identified several linguistically motivated features for discriminating arguments and their labels (see Table 1). These features usually characterize aspects of individual arguments and the predicate.

It is evident that the labels and the features of arguments are highly correlated. For example, there are hard constraints – that arguments cannot overlap with each other or the predicate, and also *soft* constraints – for example, is it unlikely that a predicate will have two or more AGENT arguments, or that a predicate used in the active voice will have a THEME argument prior to an AGENT argument. Several systems have incorporated such dependencies, for example, (Gildea and Jurafsky, 2002; Pradhan et al., 2004; Thompson et al., 2003) and several systems submitted in the CoNLL-2004 shared task (Carreras and Màrquez, 2004). However, we show that there are greater gains to be had by modeling joint information about a verb's argument structure.

We propose a discriminative log-linear joint model for semantic role labeling, which incorporates more global features and achieves superior performance in comparison to state-of-the-art models. To deal with the computational complexity of the task, we employ dynamic programming and reranking approaches. We present performance results on the February 2004 version of PropBank on gold-standard parse trees as well as results on automatic parses generated by Charniak's parser (Charniak, 2000).

2 Semantic Role Labeling: Task Definition and Architectures

Consider the pair of sentences,

- [The GM-Jaguar pact]$_{\text{AGENT}}$ *gives* [the car market]$_{\text{RECIPIENT}}$ [a much-needed boost]$_{\text{THEME}}$
- [A much-needed boost]$_{\text{THEME}}$ was *given* to [the car market]$_{\text{RECIPIENT}}$ by [the GM-Jaguar pact]$_{\text{AGENT}}$

Despite the different syntactic positions of the labeled phrases, we recognize that each plays the same

Proceedings of the 43rd Annual Meeting of the ACL, pages 589–596,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

role – indicated by the label – in the meaning of this sense of the verb *give*. We call such phrases fillers of semantic roles and our task is, given a sentence and a target verb, to return all such phrases along with their correct labels. Therefore one subtask is to group the words of a sentence into phrases or constituents. As in most previous work on semantic role labeling, we assume the existence of a separate parsing model that can assign a parse tree t to each sentence, and the task then is to label each node in the parse tree with the semantic role of the phrase it dominates, or NONE, if the phrase does not fill any role. We do stress however that the joint framework and features proposed here can also be used when only a shallow parse (chunked) representation is available as in the CoNLL-2004 shared task (Carreras and Màrquez, 2004).

In the February 2004 version of the PropBank corpus, annotations are done on top of the Penn TreeBank II parse trees (Marcus et al., 1993). Possible labels of arguments in this corpus are the *core* argument labels ARG[0-5], and the *modifier* argument labels. The core arguments ARG[3-5] do not have consistent global roles and tend to be verb specific. There are about 14 modifier labels such as ARGM-LOC and ARGM-TMP, for location and temporal modifiers respectively.[1] Figure 1 shows an example parse tree annotated with semantic roles.

We distinguish between models that learn to label nodes in the parse tree independently, called *local models*, and models that incorporate dependencies among the labels of multiple nodes, called *joint models*. We build both local and joint models for semantic role labeling, and evaluate the gains achievable by incorporating joint information. We start by introducing our local models, and later build on them to define joint models.

3 Local Classifiers

In the context of role labeling, we call a classifier local if it assigns a probability (or score) to the label of an individual parse tree node n_i independently of the labels of other nodes.

We use the standard separation of the task of semantic role labeling into *identification* and *classification* phases. In *identification*, our task is to classify nodes of t as either ARG, an argument (including modifiers), or NONE, a non-argument. In *classification*, we are given a set of arguments in t and must label each one with its appropriate semantic role. Formally, let L denote a mapping of the nodes in t to a label set of semantic roles (including NONE) and let $Id(L)$ be the mapping which collapses L's non-NONE values into ARG. Then we can decompose the probability of a labeling L into probabilities according to an identification model P_{ID} and a classification model P_{CLS}.

$$\begin{aligned} P_{SRL}(L|t,v) &= P_{ID}(Id(L)|t,v) \times \\ &\quad P_{CLS}(L|t,v,Id(L)) \end{aligned} \quad (1)$$

This decomposition does not encode any independence assumptions, but is a useful way of thinking about the problem. Our local models for semantic role labeling use this decomposition. Previous work has also made this distinction because, for example, different features have been found to be more effective for the two tasks, and it has been a good way to make training and search during testing more efficient.

Here we use the same features for local identification and classification models, but use the decomposition for efficiency of training. The identification models are trained to classify each node in a parse tree as ARG or NONE, and the classification models are trained to label each argument node in the training set with its specific label. In this way the training set for the classification models is smaller. Note that we don't do any hard pruning at the identification stage in testing and can find the exact labeling of the complete parse tree, which is the maximizer of Equation 1. Thus we do not have accuracy loss as in the two-pass hard prune strategy described in (Pradhan et al., 2005).

In previous work, various machine learning methods have been used to learn local classifiers for role labeling. Examples are linearly interpolated relative frequency models (Gildea and Jurafsky, 2002), SVMs (Pradhan et al., 2004), decision trees (Surdeanu et al., 2003), and log-linear models (Xue and Palmer, 2004). In this work we use log-linear models for multi-class classification. One advantage of log-linear models over SVMs for us is that they produce probability distributions and thus identification

[1]For a full listing of PropBank argument labels see (Palmer et al., 2003)

Standard Features (Gildea and Jurafsky, 2002)
PHRASE TYPE: Syntactic Category of node
PREDICATE LEMMA: Stemmed Verb
PATH: Path from node to predicate
POSITION: Before or after predicate?
VOICE: Active or passive relative to predicate
HEAD WORD OF PHRASE
SUB-CAT: CFG expansion of predicate's parent
Additional Features (Pradhan et al., 2004)
FIRST/LAST WORD
LEFT/RIGHT SISTER PHRASE-TYPE
LEFT/RIGHT SISTER HEAD WORD/POS
PARENT PHRASE-TYPE
PARENT POS/HEAD-WORD
ORDINAL TREE DISTANCE: Phrase Type with appended length of PATH feature
NODE-LCA PARTIAL PATH Path from constituent to Lowest Common Ancestor with predicate node
PP PARENT HEAD WORD If parent is a PP return parent's head word
PP NP HEAD WORD/POS For a PP, retrieve the head Word / POS of its rightmost NP
Selected Pairs (Xue and Palmer, 2004)
PREDICATE LEMMA & PATH
PREDICATE LEMMA & HEAD WORD
PREDICATE LEMMA & PHRASE TYPE
VOICE & POSITION
PREDICATE LEMMA & PP PARENT HEAD WORD

Table 1: Baseline Features

and classification models can be chained in a principled way, as in Equation 1.

The features we used for local identification and classification models are outlined in Table 1. These features are a subset of features used in previous work. The standard features at the top of the table were defined by (Gildea and Jurafsky, 2002), and the rest are other useful lexical and structural features identified in more recent work (Pradhan et al., 2004; Surdeanu et al., 2003; Xue and Palmer, 2004).

The most direct way to use trained local identification and classification models in testing is to select a labeling L of the parse tree that maximizes the product of the probabilities according to the two models as in Equation 1. Since these models are local, this is equivalent to independently maximizing the product of the probabilities of the two models for the label l_i of each parse tree node n_i as shown below in Equation 2.

$$\begin{aligned} P^{\ell}_{SRL}(L|t,v) &= \prod_{n_i \in t} P_{ID}(Id(l_i)|t,v) \\ &\times \prod_{n_i \in t} P_{CLS}(l_i|t,v,Id(l_i)) \end{aligned} \quad (2)$$

A problem with this approach is that a maximizing labeling of the nodes could possibly violate the constraint that argument nodes should not overlap with each other. Therefore, to produce a consistent set of arguments with local classifiers, we must have a way of enforcing the non-overlapping constraint.

3.1 Enforcing the Non-overlapping Constraint

Here we describe a fast exact dynamic programming algorithm to find the most likely non-overlapping (consistent) labeling of all nodes in the parse tree, according to a product of probabilities from local models, as in Equation 2. For simplicity, we describe the dynamic program for the case where only two classes are possible – ARG and NONE. The generalization to more classes is straightforward. Intuitively, the algorithm is similar to the Viterbi algorithm for context-free grammars, because we can describe the non-overlapping constraint by a "grammar" that disallows ARG nodes to have ARG descendants.

Below we will talk about maximizing the sum of the logs of local probabilities rather than the product of local probabilities, which is equivalent. The dynamic program works from the leaves of the tree up and finds a best assignment for each tree, using already computed assignments for its children. Suppose we want the most likely consistent assignment for subtree t with children trees $t_1, \ldots, t_k$ each storing the most likely consistent assignment of nodes it dominates as well as the log-probability of the assignment of all nodes it dominates to NONE. The most likely assignment for t is the one that corresponds to the maximum of:

- The sum of the log-probabilities of the most likely assignments of the children subtrees $t_1, \ldots, t_k$ plus the log-probability for assigning the node t to NONE

- The sum of the log-probabilities for assigning all of t_i's nodes to NONE plus the log-probability for assigning the node t to ARG.

Propagating this procedure from the leaves to the root of t, we have our most likely non-overlapping assignment. By slightly modifying this procedure, we obtain the most likely assignment according to a product of local identification and classification models. We use the local models in conjunction with this search procedure to select a most likely labeling in testing. Test set results for our local model P^{ℓ}_{SRL} are given in Table 2.

4 Joint Classifiers

As discussed in previous work, there are strong dependencies among the labels of the semantic argument nodes of a verb. A drawback of local models is that, when they decide the label of a parse tree node, they cannot use information about the labels and features of other nodes in the tree.

Furthermore, these dependencies are highly non-local. For instance, to avoid repeating argument labels in a frame, we need to add a dependency from each node label to the labels of all other nodes. A factorized sequence model that assumes a finite Markov horizon, such as a chain Conditional Random Field (Lafferty et al., 2001), would not be able to encode such dependencies.

The need for Re-ranking

For argument identification, the number of possible assignments for a parse tree with n nodes is 2^n. This number can run into the hundreds of billions for a normal-sized tree. For argument labeling, the number of possible assignments is $\approx 20^m$, if m is the number of arguments of a verb (typically between 2 and 5), and 20 is the approximate number of possible labels if considering both core and modifying arguments. Training a model which has such huge number of classes is infeasible if the model does not factorize due to strong independence assumptions. Therefore, in order to be able to incorporate long-range dependencies in our models, we chose to adopt a re-ranking approach (Collins, 2000), which selects from likely assignments generated by a model which makes stronger independence assumptions. We utilize the top N assignments of our local semantic role labeling model P^{ℓ}_{SRL} to generate likely assignments. As can be seen from Table 3, for relatively small values of N, our re-ranking approach does not present a serious bottleneck to performance. We used a value of $N = 20$ for training. In Table 3 we can see that if we could pick, using an oracle, the best assignment out for the top 20 assignments according to the local model, we would achieve an F-Measure of 98.8 on all arguments. Increasing the number of N to 30 results in a very small gain in the upper bound on performance and a large increase in memory requirements. We therefore selected $N = 20$ as a good compromise.

Generation of top N most likely joint assignments

We generate the top N most likely non-overlapping joint assignments of labels to nodes in a parse tree according to a local model P^{ℓ}_{SRL}, by an exact dynamic programming algorithm, which is a generalization of the algorithm for finding the top non-overlapping assignment described in section 3.1.

Parametric Models

We learn log-linear re-ranking models for joint semantic role labeling, which use feature maps from a parse tree and label sequence to a vector space. The form of the models is as follows. Let $\Phi(t, v, L) \in \mathbb{R}^s$ denote a feature map from a tree t, target verb v, and joint assignment L of the nodes of the tree, to the vector space $\mathbb{R}^s$. Let $L_1, L_2, \cdots, L_N$ denote top N possible joint assignments. We learn a log-linear model with a parameter vector W, with one weight for each of the s dimensions of the feature vector. The probability (or score) of an assignment L according to this re-ranking model is defined as:

$$P^{r}_{SRL}(L|t, v) = \frac{e^{\langle \Phi(t,v,L), W \rangle}}{\sum_{j=1}^{N} e^{\langle \Phi(t,v,L_j).W \rangle}} \tag{3}$$

The score of an assignment L not in the top N is zero. We train the model to maximize the sum of log-likelihoods of the best assignments minus a quadratic regularization term.

In this framework, we can define arbitrary features of labeled trees that capture general properties of predicate-argument structure.

Joint Model Features

We will introduce the features of the joint re-ranking model in the context of the example parse tree shown in Figure 1. We model dependencies not only between the label of a node and the labels of

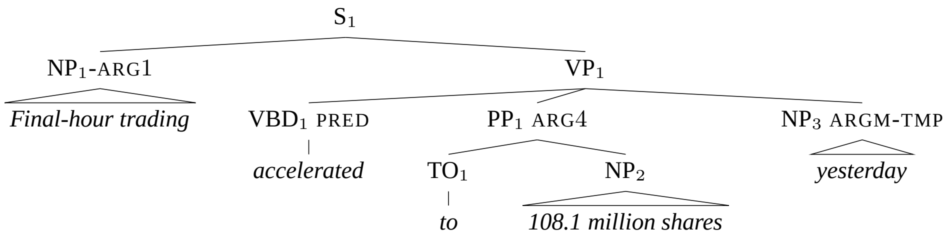


Figure 1: An example tree from the PropBank with Semantic Role Annotations.

other nodes, but also dependencies between the label of a node and input features of other argument nodes. The features are specified by instantiation of templates and the value of a feature is the number of times a particular pattern occurs in the labeled tree.

Templates

For a tree t, predicate v, and joint assignment L of labels to the nodes of the tree, we define the *candidate argument sequence* as the sequence of non-NONE labeled nodes $[n_1, l_1, \ldots, v_{PRED}, n_m, l_m]$ (l_i is the label of node n_i). A reasonable candidate argument sequence usually contains very few of the nodes in the tree – about 2 to 7 nodes, as this is the typical number of arguments for a verb. To make it more convenient to express our feature templates, we include the predicate node v in the sequence. This sequence of labeled nodes is defined with respect to the left-to-right order of constituents in the parse tree. Since non-NONE labeled nodes do not overlap, there is a strict left-to-right order among these nodes. The candidate argument sequence that corresponds to the correct assignment in Figure 1 will be:

[NP_1-ARG1,VBD_1-PRED,PP_1-ARG4,NP_3-ARGM-TMP]

Features from Local Models: All features included in the local models are also included in our joint models. In particular, each template for local features is included as a joint template that concatenates the local template and the node label. For example, for the local feature PATH, we define a joint feature template, that extracts PATH from every node in the candidate argument sequence and concatenates it with the label of the node. Both a feature with the specific argument label is created and a feature with the generic back-off ARG label. This is similar to adding features from identification and classification models. In the case of the example candidate argument sequence above, for the node NP_1 we have the features:

(NP↑S↓)-ARG1, (NP↑S↓)-ARG

When comparing a local and a joint model, we use the same set of local feature templates in the two models.

Whole Label Sequence: As observed in previous work (Gildea and Jurafsky, 2002; Pradhan et al., 2004), including information about the set or sequence of labels assigned to argument nodes should be very helpful for disambiguation. For example, including such information will make the model less likely to pick multiple fillers for the same role or to come up with a labeling that does not contain an obligatory argument. We added a whole label sequence feature template that extracts the labels of all argument nodes, and preserves information about the position of the predicate. The template also includes information about the voice of the predicate. For example, this template will be instantiated as follows for the example candidate argument sequence:

[voice:active ARG1,PRED,ARG4,ARGM-TMP]

We also add a variant of this feature which uses a generic ARG label instead of specific labels. This feature template has the effect of counting the number of arguments to the left and right of the predicate, which provides useful global information about argument structure. As previously observed (Pradhan et al., 2004), including modifying arguments in sequence features is not helpful. This was confirmed in our experiments and we redefined the whole label sequence features to exclude modifying arguments.

One important variation of this feature uses the actual predicate lemma in addition to “voice:active”. Additionally, we define variations of these feature templates that concatenate the label sequence with features of individual nodes. We experimented with variations, and found that including the phrase type and the head of a directly dominating PP – if one exists – was most helpful. We also add a feature that detects repetitions of the same label in a candidate argument sequence, together with the phrase types of the nodes labeled with that label. For example, (NP-ARG0,WHNP-ARG0) is a common pattern of this form.

Frame Features: Another very effective class of features we defined are features that look at the label of a single argument node and internal features of other argument nodes. The idea of these features is to capture knowledge about the label of a constituent given the syntactic realization of all arguments of the verb. This is helpful to capture syntactic alternations, such as the dative alternation. For example, consider the sentence (*i*) "[Shaw Publishing]$_{\text{ARG}_0}$ *offered* [Mr. Smith]$_{\text{ARG2}}$ [a reimbursement]$_{\text{ARG}_1}$" and the alternative realization (*ii*) "[Shaw Publishing]$_{\text{ARG}_0}$ *offered* [a reimbursement]$_{\text{ARG}_1}$ [to Mr. Smith]$_{\text{ARG}_2}$". When classifying the NP in object position, it is useful to know whether the following argument is a PP. If yes, the NP will more likely be an ARG$_1$, and if not, it will more likely be an ARG$_2$. A feature template that captures such information extracts, for each argument node, its phrase type and label in the context of the phrase types for all other arguments. For example, the instantiation of such a template for [a reimbursement] in (*ii*) would be

[voice:active NP,PRED,NP-ARG$_1$,PP]

We also add a template that concatenates the identity of the predicate lemma itself.

We should note that Xue and Palmer (2004) define a similar feature template, called *syntactic frame*, which often captures similar information. The important difference is that their template extracts contextual information from noun phrases surrounding the predicate, rather than from the sequence of argument nodes. Because our model is joint, we are able to use information about other argument nodes when labeling a node.

Final Pipeline

Here we describe the application in testing of a joint model for semantic role labeling, using a local model P^{ℓ}_{SRL}, and a joint re-ranking model P^{r}_{SRL}. P^{ℓ}_{SRL} is used to generate top N non-overlapping joint assignments $L_1, \ldots, L_N$.

One option is to select the best L_i according to P^{r}_{SRL}, as in Equation 3, ignoring the score from the local model. In our experiments, we noticed that for larger values of N, the performance of our re-ranking model P^{r}_{SRL} decreased. This was probably due to the fact that at test time the local classifier produces very poor argument frames near the bottom of the top N for large N. Since the re-ranking model is trained on relatively few good argument frames, it cannot easily rule out very bad frames. It makes sense then to incorporate the local model into our final score. Our final score is given by:

$$P_{SRL}(L|t,v) = (P^{\ell}_{SRL}(L|t,v))^{\alpha}\, P^{r}_{SRL}(L|t,v)$$

where α is a tunable parameter [2] for how much influence the local score has in the final score. Such interpolation with a score from a first-pass model was also used for parse re-ranking in (Collins, 2000). Given this score, at test time we choose among the top N local assignments $L_1, \ldots, L_N$ according to:

$$\underset{L\in\{L_1,\ldots,L_N\}}{\arg\max}\ \alpha \log P^{\ell}_{SRL}(L|t,v) + \log P^{r}_{SRL}(L|t,v)$$

5 Experiments and Results

For our experiments we used the February 2004 release of PropBank. [3] As is standard, we used the annotations from sections 02–21 for training, 24 for development, and 23 for testing. As is done in some previous work on semantic role labeling, we discard the relatively infrequent discontinuous arguments from both the training and test sets. In addition to reporting the standard results on individual argument F-Measure, we also report Frame Accuracy (Acc.), the fraction of sentences for which we successfully label all nodes. There are reasons to prefer Frame Accuracy as a measure of performance over individual-argument statistics. Foremost, potential applications of role labeling may require correct labeling of all (or at least the core) arguments in a sentence in order to be effective, and partially correct labelings may not be very useful.

[2] We found $\alpha = 0.5$ to work best

[3] Although the fi rst offi cial release of PropBank was recently released, we have not had time to test on it.

Task	CORE		ARGM	
	F1	Acc.	F1	Acc.
Identifi cation	95.1	84.0	95.2	80.5
Classifi cation	96.0	93.3	93.6	85.6
Id+Classifi cation	92.2	80.7	89.9	71.8

Table 2: Performance of local classifiers on identification, classification, and identification+classification on section 23, using gold-standard parse trees.

N	CORE		ARGM	
	F1	Acc.	F1	Acc.
1	92.2	80.7	89.9	71.8
5	97.8	93.9	96.8	89.5
20	99.2	97.4	98.8	95.3
30	99.3	97.9	99.0	96.2

Table 3: Oracle upper bounds for performance on the complete identification+classification task, using varying numbers of top N joint labelings according to local classifiers.

Model	CORE		ARGM	
	F1	Acc.	F1	Acc.
Local	92.2	80.7	89.9	71.8
Joint	**94.7**	**88.2**	**92.1**	**79.4**

Table 4: Performance of local and joint models on identification+classification on section 23, using gold-standard parse trees.

We report results for two variations of the semantic role labeling task. For CORE, we identify and label only core arguments. For ARGM, we identify and label core as well as modifier arguments. We report results for local and joint models on argument identification, argument classification, and the complete identification and classification pipeline. Our local models use the features listed in Table 1 and the technique for enforcing the non-overlapping constraint discussed in Section 3.1.

The labeling of the tree in Figure 1 is a specific example of the kind of errors fixed by the joint models. The local classifier labeled the first argument in the tree as ARG0 instead of ARG1, probably because an ARG0 label is more likely for the subject position.

All joint models for these experiments used the whole sequence and frame features. As can be seen from Table 4, our joint models achieve error reductions of 32% and 22% over our local models in F-Measure on CORE and ARGM respectively. With respect to the Frame Accuracy metric, the joint error reduction is 38% and 26% for CORE and ARGM respectively.

We also report results on automatic parses (see Table 5). We trained and tested on automatic parse trees from Charniak's parser (Charniak, 2000). For approximately 5.6% of the argument constituents in the test set, we could not find exact matches in the automatic parses. Instead of discarding these arguments, we took the largest constituent in the automatic parse having the same head-word as the gold-standard argument constituent. Also, 19 of the propositions in the test set were discarded because Charniak's parser altered the tokenization of the input sentence and tokens could not be aligned. As our results show, the error reduction of our joint model with respect to the local model is more modest in this setting. One reason for this is the lower upper bound, due largely to the the much poorer performance of the identification model on automatic parses. For ARGM, the local identification model achieves 85.9 F-Measure and 59.4 Frame Accuracy; the local classification model achieves 92.3 F-Measure and 83.1 Frame Accuracy. It seems that the largest boost would come from features that can identify arguments in the presence of parser errors, rather than the features of our joint model, which ensure global coherence of the argument frame. We still achieve 10.7% and 18.5% error reduction for CORE arguments in F-Measure and Frame Accuracy respectively.

Model	CORE		ARGM	
	F1	Acc.	F1	Acc.
Local	84.1	66.5	81.4	55.6
Joint	85.8	72.7	82.9	60.8

Table 5: Performance of local and joint models on identification+classification on section 23, using Charniak automatically generated parse trees.

6 Related Work

Several semantic role labeling systems have successfully utilized joint information. (Gildea and Jurafsky, 2002) used the empirical probability of the set of proposed arguments as a prior distribution. (Pradhan et al., 2004) train a language model over label sequences. (Punyakanok et al., 2004) use a linear programming framework to ensure that the only argument frames which get probability mass are ones that respect global constraints on argument labels.

The key differences of our approach compared to previous work are that our model has all of the following properties: (*i*) we do not assume a finite Markov horizon for dependencies among node labels, (*ii*) we include features looking at the labels of multiple argument nodes and *internal features* of these nodes, and (*iii*) we train a discriminative model capable of incorporating these long-distance dependencies.

7 Conclusions

Reflecting linguistic intuition and in line with current work, we have shown that there are substantial gains to be had by jointly modeling the argument frames of verbs. This is especially true when we model the dependencies with discriminative models capable of incorporating long-distance features.

8 Acknowledgements

The authors would like to thank the reviewers for their helpful comments and Dan Jurafsky for his insightful suggestions and useful discussions. This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program.

References

Collin Baker, Charles Fillmore, and John Lowe. 1998. The Berkeley Framenet project. In *Proceedings of COLING-ACL-1998*.

Xavier Carreras and Luís M`arquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML-2000*.

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

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fi elds: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-2001*.

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.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2003. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL-2004*.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. 2005. Support vector learning for semantic argument classifi cation. *Machine Learning Journal*.

Vasin Punyakanok, Dan Roth, Wen tau Yih, Dav Zimak, and Yuancheng Tu. 2004. Semantic role labeling via generalized inference over classifi ers. In *Proceedings of CoNLL-2004*.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL-2003*.

Cynthia A. Thompson, Roger Levy, and Christopher D. Manning. 2003. A generative model for semantic role labeling. In *Proceedings of ECML-2003*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*.

Paraphrasing with Bilingual Parallel Corpora

Colin Bannard **Chris Callison-Burch**
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh, EH8 9LW
{c.j.bannard, callison-burch}@ed.ac.uk

Abstract

Previous work has used monolingual parallel corpora to extract and generate paraphrases. We show that this task can be done using bilingual parallel corpora, a much more commonly available resource. Using alignment techniques from phrase-based statistical machine translation, we show how paraphrases in one language can be identified using a phrase in another language as a pivot. We define a paraphrase probability that allows paraphrases extracted from a bilingual parallel corpus to be ranked using translation probabilities, and show how it can be refined to take contextual information into account. We evaluate our paraphrase extraction and ranking methods using a set of manual word alignments, and contrast the quality with paraphrases extracted from automatic alignments.

1 Introduction

Paraphrases are alternative ways of conveying the same information. Paraphrases are useful in a number of NLP applications. In natural language generation the production of paraphrases allows for the creation of more varied and fluent text (Iordanskaja et al., 1991). In multidocument summarization the identification of paraphrases allows information repeated across documents to be condensed (McKeown et al., 2002). In the automatic evaluation of machine translation, paraphrases may help to alleviate problems presented by the fact that there are often alternative and equally valid ways of translating a text (Pang et al., 2003). In question answering, discovering paraphrased answers may provide additional evidence that an answer is correct (Ibrahim et al., 2003).

In this paper we introduce a novel method for extracting paraphrases that uses bilingual parallel corpora. Past work (Barzilay and McKeown, 2001; Barzilay and Lee, 2003; Pang et al., 2003; Ibrahim et al., 2003) has examined the use of monolingual parallel corpora for paraphrase extraction. Examples of monolingual parallel corpora that have been used are multiple translations of classical French novels into English, and data created for machine translation evaluation methods such as Bleu (Papineni et al., 2002) which use multiple reference translations.

While the results reported for these methods are impressive, their usefulness is limited by the scarcity of monolingual parallel corpora. Small data sets mean a limited number of paraphrases can be extracted. Furthermore, the narrow range of text genres available for monolingual parallel corpora limits the range of contexts in which the paraphrases can be used.

Instead of relying on scarce monolingual parallel data, our method utilizes the abundance of bilingual parallel data that is available. This allows us to create a much larger inventory of phrases that is applicable to a wider range of texts.

Our method for identifying paraphrases is an extension of recent work in phrase-based statistical machine translation (Koehn et al., 2003). The essence of our method is to align phrases in a bilingual parallel corpus, and equate different English phrases that are aligned with the same phrase in the other language. This assumption of similar mean-

Emma burst into tears **and he tried to** comfort **her,** saying things to make her smile.
Emma cried, **and he tried to** console **her,** adorning his words with puns.

Figure 1: Using a monolingal parallel corpus to extract paraphrases

ing when multiple phrases map onto a single foreign language phrase is the converse of the assumption made in the word sense disambiguation work of Diab and Resnik (2002) which posits different word senses when a single English word maps onto different words in the foreign language (we return to this point in Section 4.4).

The remainder of this paper is as follows: Section 2 contrasts our method for extracting paraphrases with the monolingual case, and describes how we rank the extracted paraphrases with a probability assignment. Section 3 describes our experimental setup and includes information about how phrases were selected, how we manually aligned parts of the bilingual corpus, and how we evaluated the paraphrases. Section 4 gives the results of our evaluation and gives a number of example paraphrases extracted with our technique. Section 5 reviews related work, and Section 6 discusses future directions.

2 Extracting paraphrases

Much previous work on extracting paraphrases (Barzilay and McKeown, 2001; Barzilay and Lee, 2003; Pang et al., 2003) has focused on finding identifying contexts within aligned monolingual sentences from which divergent text can be extracted, and treated as paraphrases. Barzilay and McKeown (2001) gives the example shown in Figure 1 of how identical surrounding substrings can be used to extract the paraphrases of *burst into tears* as *cried* and *comfort* as *console*.

While monolingual parallel corpora often have identical contexts that can be used for identifying paraphrases, bilingual parallel corpora do not. Instead, we use phrases in the other language as pivots: we look at what foreign language phrases the English translates to, find all occurrences of those foreign phrases, and then look back at what other English phrases they translate to. We treat the other English phrases as potential paraphrases. Figure 2 illustrates how a German phrase can be used as a point of identification for English paraphrases in this way. Section 2.1 explains which statistical machine translation techniques are used to align phrases within sentence pairs in a bilingual corpus.

A significant difference between the present work and that employing monolingual parallel corpora, is that our method frequently extracts more than one possible paraphrase for each phrase. We assign a probability to each of the possible paraphrases. This is a mechanism for ranking paraphrases, which can be utilized when we come to select the correct paraphrase for a given context . Section 2.2 explains how we calculate the probability of a paraphrase.

2.1 Aligning phrase pairs

We use phrase alignments in a parallel corpus as pivots between English paraphrases. We find these alignments using recent *phrase-based* approaches to statistical machine translation.

The original formulation of statistical machine translation (Brown et al., 1993) was defined as a word-based operation. The probability that a foreign sentence is the translation of an English sentence is calculated by summing over the probabilities of all possible word-level alignments, $\mathbf{a}$, between the sentences:

$$p(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

Thus Brown et al. decompose the problem of determining whether a sentence is a good translation of another into the problem of determining whether there is a sensible mapping between the words in the sentences.

More recent approaches to statistical translation calculate the translation probability using larger blocks of aligned text. Koehn (2004), Tillmann (2003), and Vogel et al. (2003) describe various heuristics for extracting phrase alignments from the Viterbi word-level alignments that are estimated using Brown et al. (1993) models. We use the heuristic for phrase alignment described in Och and Ney (2003) which aligns phrases by incrementally building longer phrases from words and phrases which have adjacent alignment points.[1]

[1]Note that while we induce the translations of phrases from

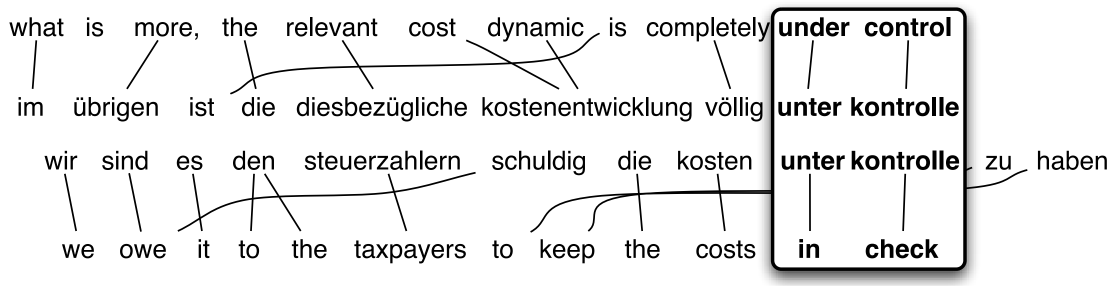

what is more, the relevant cost dynamic is completely under control
im übrigen ist die diesbezügliche kostenentwicklung völlig unter kontrolle
wir sind es den steuerzahlern schuldig die kosten unter kontrolle zu haben
we owe it to the taxpayers to keep the costs in check

Figure 2: Using a bilingual parallel corpus to extract paraphrases

2.2 Assigning probabilities

We define a paraphrase probability $p(e_2|e_1)$ in terms of the translation model probabilities $p(f|e_1)$, that the original English phrase e_1 translates as a particular phrase f in the other language, and $p(e_2|f)$, that the candidate paraphrase e_2 translates as the foreign language phrase. Since e_1 can translate as multiple foreign language phrases, we sum over f:

$$\hat{e_2} = \arg\max_{e_2 \neq e_1} p(e_2|e_1) \quad (1)$$

$$= \arg\max_{e_2 \neq e_1} \sum_f p(f|e_1)p(e_2|f) \quad (2)$$

The translation model probabilities can be computed using any standard formulation from phrase-based machine translation. For example, $p(e|f)$ can be calculated straightforwardly using maximum likelihood estimation by counting how often the phrases e and f were aligned in the parallel corpus:

$$p(e|f) = \frac{count(e,f)}{\sum_e count(e,f)} \quad (3)$$

Note that the paraphrase probability defined in Equation 2 returns the single best paraphrase, $\hat{e_2}$, irrespective of the context in which e_1 appears. Since the best paraphrase may vary depending on information about the sentence that e_1 appears in, we extend the paraphrase probability to include that sentence S:

$$\hat{e_2} = \arg\max_{e_2 \neq e_1} p(e_2|e_1, S) \quad (4)$$

word-level alignments in this paper, direct estimation of phrasal translations (Marcu and Wong, 2002) would also suffice for extracting paraphrases from bilingual corpora.

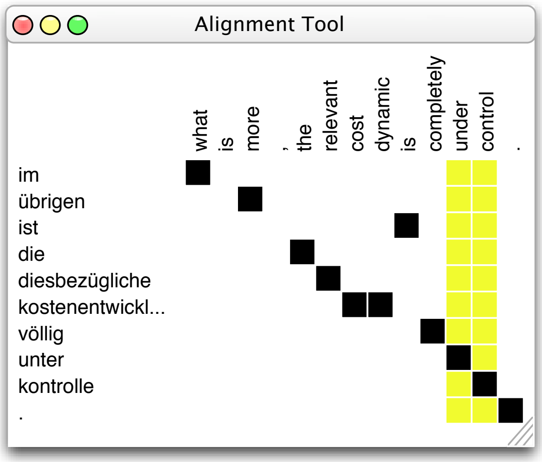
a million, as far as possible, at work, big business, carbon dioxide, central america, close to, concentrate on, crystal clear, do justice to, driving force, first half, for the first time, global warming, great care, green light, hard core, horn of africa, last resort, long ago, long run, military action, military force, moment of truth, new world, noise pollution, not to mention, nuclear power, on average, only too, other than, pick up, president clinton, public transport, quest for, red cross, red tape, socialist party, sooner or later, step up, task force, turn to, under control, vocational training, western sahara, world bank

Table 1: Phrases that were selected to paraphrase

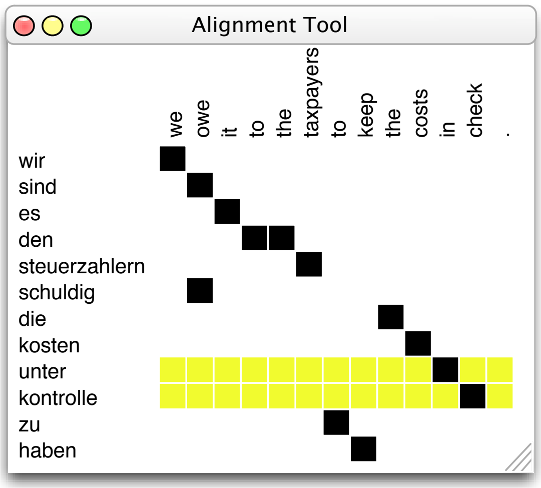
S allows us to re-rank the candidate paraphrases based on additional contextual information. The experiments in this paper employ one variety of contextual information. We include a simple language model probability, which would additionally rank e_2 based on the probability of the sentence formed by substiuting e_2 for e_1 in S. A possible extension which we do not evaluate might be permitting only paraphrases that are the same syntactic type as the original phrase, which we could do by extending the translation model probabilities to count only phrase occurrences of that type.

3 Experimental Design

We extracted 46 English phrases to paraphrase (shown in Table 1), randomly selected from those multi-word phrases in WordNet which also occured multiple times in the first 50,000 sentences of our bilingual corpus. The bilingual corpus that we used



(a) Aligning the English phrase to be paraphrased



(b) Aligning occurrences of its German translation

Figure 3: Phrases highlighted for manual alignment

was the German-English section of the Europarl corpus, version 2 (Koehn, 2002). We produced automatic alignments for it with the Giza++ toolkit (Och and Ney, 2003). Because we wanted to test our method independently of the quality of word alignment algorithms, we also developed a gold standard of word alignments for the set of phrases that we wanted to paraphrase.

3.1 Manual alignment

The gold standard alignments were created by highlighting all occurrences of the English phrase to paraphrase and manually aligning it with its German equivalent by correcting the automatic alignment, as shown in Figure 3a. All occurrences of its German equivalents were then highlighted, and aligned with their English translations (Figure 3b). The other words in the sentences were left with their automatic alignments.

3.2 Paraphrase evaluation

We evaluated the accuracy of each of the paraphrases that was extracted from the manually aligned data, as well as the top ranked paraphrases from the experimental conditions detailed below in Section 3.3. Because the acccuracy of paraphrases can vary depending on context, we substituted each

Under control
This situation is **in check** in terms of security.
This situation is **checked** in terms of security.
This situation is **curbed** in terms of security.
This situation is **curb** in terms of security.
This situation is **limit** in terms of security.
This situation is **slow down** in terms of security.

Figure 4: Paraphrases substituted in for the original phrase

set of candidate paraphrases into between 2–10 sentences which contained the original phrase. Figure 4 shows the paraphrases for *under control* substituted into one of the sentences in which it occurred. We created a total of 289 such evaluation sets, with a total of 1366 unique sentences created through substitution.

We had two native English speakers produce judgments as to whether the new sentences preserved the meaning of the original phrase and as to whether they remained grammatical. Paraphrases that were judged to preserve both meaning and grammaticality were considered to be correct, and examples which failed on either judgment were considered to be incorrect.

In Figure 4 *in check*, *checked*, and *curbed* were

under control	checked, curb, curbed, *in check*, limit, slow down
sooner or later	*at some point*, eventually
military force	armed forces, defence, *force*, forces, military forces, peace-keeping personnel
long ago	a little time ago, a long time, *a long time ago*, a lot of time, a while ago, a while back, far, for a long time, for some time, for such a long time, long, long period of time, long term, long time, long while, overdue, some time, some time ago
green light	approval, call, *go-ahead*, indication, message, sign, signal, signals, formal go-ahead
great care	a careful approach, greater emphasis, *particular attention*, special attention, specific attention, very careful
first half	*first six months*
crystal clear	absolutely clear, all clarity, clear, clearly, in great detail, no mistake, no uncertain, obvious, obviously, particularly clear, perfectly clear, quite clear, quite clearly, quite explicitly, quite openly, very clear, *very clear and comprehensive*, very clearly, very sure, very unclear, very well
carbon dioxide	*co2*
at work	at the workplace, employment, held, holding, in the work sphere, operate, organised, taken place, took place, *working*

Table 2: Paraphrases extracted from a manually word-aligned parallel corpus

judged to be correct and *curb*, *limit* and *slow down* were judged to be incorrect. The inter-annotator agreement for these judgements was measured at $\kappa = 0.605$, which is conventionally interpreted as "good" agreement.

3.3 Experiments

We evaluated the accuracy of top ranked paraphrases when the paraphrase probability was calculated using:

1. The manual alignments,
2. The automatic alignments,
3. Automatic alignments produced over multiple corpora in different languages,
4. All of the above with language model re-ranking.
5. All of the above with the candidate paraphrases limited to the same sense as the original phrase.

4 Results

We report the percentage of correct translations (accuracy) for each of these experimental conditions. A summary of these can be seen in Table 3. This section will describe each of the set-ups and the score reported in more detail.

4.1 Manual alignments

Table 2 gives a set of example paraphrases extracted from the gold standard alignments. The italicized paraphrases are those that were assigned the highest probability by Equation 2, which chooses a single best paraphrase without regard for context. The 289 sentences created by substituting the italicized paraphrases in for the original phrase were judged to be correct an average of 74.9% of the time.

Ignoring the constraint that the new sentences remain grammatically correct, these paraphrases were judged to have the correct meaning 84.7% of the time. This suggests that the context plays a more important role with respect to the grammaticality of substituted paraphrases than with respect to their meaning.

In order to allow the surrounding words in the sentence to have an influence on which paraphrase was selected, we re-ranked the paraphrase probabilities based on a trigram language model trained on the entire English portion of the Europarl corpus. Paraphrases were selected from among all those in Table 2, and not constrained to the italicized phrases. In the case of the paraphrases extracted from the manual word alignments, the language model re-ranking had virtually no influence, and resulted in a slight dip in accuracy to 71.7%

	Paraphrase Prob	Paraphrase Prob & LM	Correct Meaning
Manual Alignments	74.9	71.7	84.7
Automatic Alignments	48.9	55.3	64.5
Using Multiple Corpora	55.0	57.4	65.4
Word Sense Controlled	57.0	61.9	70.4

Table 3: Paraphrase accuracy and correct meaning for the different data conditions

4.2 Automatic alignments

In this experimental condition paraphrases were extracted from a set of automatic alignments produced by running Giza++ over a set of 1,036,000 German-English sentence pairs (roughly 28,000,000 words in each language). When the single best paraphrase (irrespective of context) was used in place of the original phrase in the evaluation sentence the accuracy reached 48.9% which is quite low compared to the 74.9% of the manually aligned set.

As with the manual alignments it seems that we are selecting phrases which have the correct meaning but are not grammatical in context. Indeed our judges thought the meaning of the paraphrases to be correct in 64.5% of cases. Using a language model to select the best paraphrase given the context reduces the number of ungrammatical examples and gives an improvement in quality from 48.9% to 55.3% correct.

These results suggest two things: that improving the quality of automatic alignments would lead to more accurate paraphrases, and that there is room for improvement in limiting the paraphrases by their context. We address these points below.

4.3 Using multiple corpora

Work in statistical machine translation suggests that, like many other machine learning problems, performance increases as the amount of training data increases. Och and Ney (2003) show that the accuracy of alignments produced by Giza++ improve as the size of the training corpus increases.

Since we used the whole of the German-English section of the Europarl corpus, we could not try improving the alignments by simply adding more German-English training data. However, there is nothing that limits our paraphrase extraction method to drawing on candidate paraphrases from a single target language. We therefore re-formulated the paraphrase probability to include multiple corpora, as follows:

$$\hat{e_2} = \arg\max_{e_2 \neq e_1} \sum_{C} \sum_{f \; in \; C} p(f|e_1)p(e_2|f) \quad (5)$$

where C is a parallel corpus from a set of parallel corpora.

For this condition we used Giza++ to align the French-English, Spanish-English, and Italian-English portions of the Europarl corpus in addition to the German-English portion, for a total of around 4,000,000 sentence pairs in the training data.

The accuracy of paraphrases extracted over multiple corpora increased to 55%, and further to 57.4% when the language model re-ranking was included.

4.4 Controlling for word sense

As mentioned in Section 1, the way that we extract paraphrases is the converse of the methodology employed in word sense disambiguation work that uses parallel corpora (Diab and Resnik, 2002). The assumption made in the word sense disambiguation work is that if a source language word aligns with different target language words then those words may represent different word senses. This can be observed in the paraphrases for *at work* in Table 2. The paraphrases *at the workplace*, *employment*, and *in the work sphere* are a different sense of the phrase than *operate*, *held*, and *holding*, and they are aligned with different German phrases.

When we calculate the paraphrase probability we sum over different target language phrases. Therefore the English phrases that are aligned with the different German phrases (which themselves maybe indicative of different word senses) are mingled. Performance may be degraded since paraphrases that reflect different senses of the original phrase, and which therefore have a different meaning, are included in the same candidate set.

We therefore performed an experiment to see whether improvement could be had by limiting the candidate paraphrases to be the same sense as the original phrase in each test sentence. To do this, we used the fact that our test sentences were drawn from a parallel corpus. We limited phrases to the same word sense by constraining the candidate paraphrases to those that aligned with the same target language phrase. Our basic paraphrase calculation was therefore:

$$p(e_2|e_1, f) = p(f|e_1)p(e_2|f) \qquad (6)$$

Using the foreign language phrase to identify the word sense is obviously not applicable in monolingual settings, but acts as a convenient stand-in for a proper word sense disambiguation algorithm here.

When word sense is controlled in this way, the accuracy of the paraphrases extracted from the automatic alignments raises dramatically from 48.9% to 57% without language model re-ranking, and further to 61.9% when language model re-ranking was included.

5 Related Work

Barzilay and McKeown (2001) extract both single- and multiple-word paraphrases from a monolingual parallel corpus. They co-train a classifier to identify whether two phrases were paraphrases of each other based on their surrounding context. Two disadvantages of this method are that it requires identical bounding substrings, and has bias towards single words. For an evaluation set of 500 paraphrases, they report an average precision of 86% at identifying paraphrases out of context, and of 91% when the paraphrases are substituted into the original context of the aligned sentence. The results of our systems are not directly comparable, since Barzilay and McKeown (2001) evaluated their paraphrases with a different set of criteria (they asked judges whether to judge paraphrases based on “approximate conceptual equivalence”). Furthermore, their evaluation was carried out only by substituting the paraphrase in for the phrase with the identical context, and not in for arbitrary occurrences of the original phrase, as we have done.

Lin and Pantel (2001) use a standard (non-parallel) monolingual corpus to generate paraphrases, based on dependancy graphs and distributional similarity. One strong disadvantage of this method is that their paraphrases can also have opposite meanings.

Ibrahim et al. (2003) combine the two approaches: aligned monolingual corpora and parsing. They evaluated their system with human judges who were asked whether the paraphrases were “roughly interchangeable given the genre”, scored an average of 41% on a set of 130 paraphrases, with the judges all agreeing 75% of the time, and a correlation of 0.66. The shortcomings of this method are that it is dependent upon parse quality, and is limited by the rareness of the data.

Pang et al. (2003) use parse trees over sentences in monolingual parallel corpus to identify paraphrases by grouping similar syntactic constituents. They use heuristics such as keyword checking to limit the over-application of this method. Our alignment method might be an improvement of their heuristics for choosing which constituents ought to be grouped.

6 Discussion and Future Work

In this paper we have introduced a novel method for extracting paraphrases, which we believe greatly increases the usefulness of paraphrasing in NLP applications. The advantages of our method are that it:

- Produces a ranked list of high quality paraphrases with associated probabilities, from which the best paraphrase can be chosen according to the target context. We have shown how a language model can be used to select the best paraphrase for a particular context from this list.

- Straightforwardly handles multi-word units. Whereas for previous approaches the evaluation has been performed over mostly single word paraphrases, our results are reported exclusively over units of between 2 and 4 words.

- Because we use a much more abundant source of data, our method can be used for a much wider range of text genres than previous approaches, namely any for which parallel data is available.

One crucial thing to note is that we have demonstrated our paraphrases to be of higher quality when the alignments used to produce them are improved. This means that our method will reap the benefits of research that improvements to automatic alignment techniques (Callison-Burch et al., 2004), and will further improve as more parallel data becomes available.

In the future we plan to:

- Investigate whether our re-ranking can be further improved by using a syntax-based language model.
- Formulate a paraphrase probability for sentential paraphrases, and use this to try to identify paraphrases across documents in order to condense information for multi-document summarization.
- See whether paraphrases can be used to increase coverage for statistical machine translation when translating into "low-density" languages which have small parallel corpora.

Acknowledgments

The authors would like to thank Beatrice Alex, Marco Kuhlmann, and Josh Schroeder for their valuable input as well as their time spent annotating and contributing to the software.

References

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT/NAACL*.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*.

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

Chris Callison-Burch, David Talbot, and Miles Osborne. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proceedings of ACL*.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of ACL*.

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the Second International Workshop on Paraphrasing (ACL 2003)*.

Lidija Iordanskaja, Richard Kittredge, and Alain Polgére. 1991. Lexical selection and paraphrase in a meaning-text generation model. In Cécile L. Paris, William R. Swartout, and William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*.

Philipp Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation. Unpublished Draft.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*.

Dekang Lin and Patrick Pantel. 2001. DIRT - discovery of inference rules from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*.

Kathleen R. McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L. Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. 2002. Tracking and summarizing news on a daily basis with Columbia's Newsblaster. In *Proceedings of the Human Language Technology Conference*.

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

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT/NAACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Christoph Tillmann. 2003. A projection extension algorithm for statistical machine translation. In *Proceedings of EMNLP*.

Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. 2003. The CMU statistical machine translation system. In *Proceedings of MT Summit 9*.

A Nonparametric Method for Extraction of Candidate Phrasal Terms

Paul Deane
Center for Assessment, Design and Scoring
Educational Testing Service
pdeane@ets.org

Abstract

This paper introduces a new method for identifying candidate phrasal terms (also known as multiword units) which applies a nonparametric, rank-based heuristic measure. Evaluation of this measure, the *mutual rank ratio* metric, shows that it produces better results than standard statistical measures when applied to this task.

1 Introduction

The ordinary vocabulary of a language like English contains thousands of *phrasal terms* -- multiword lexical units including compound nouns, technical terms, idioms, and fixed collocations. The exact number of phrasal terms is difficult to determine, as new ones are coined regularly, and it is sometimes difficult to determine whether a phrase is a fixed term or a regular, compositional expression. Accurate identification of phrasal terms is important in a variety of contexts, including natural language parsing, question answering systems, information retrieval systems, among others.

Insofar as phrasal terms function as lexical units, their component words tend to cooccur more often, to resist substitution or paraphrase, to follow fixed syntactic patterns, and to display some degree of semantic noncompositionality (Manning, 1999:183-186). However, none of these characteristics are amenable to a simple algorithmic interpretation. It is true that various term extraction systems have been developed, such as Xtract (Smadja 1993), Termight (Dagan & Church 1994), and TERMS (Justeson & Katz 1995) among others (cf. Daille 1996, Jacquemin & Tzoukermann 1994, Jacquemin, Klavans, & Toukermann 1997, Boguraev & Kennedy 1999, Lin 2001). Such systems typically rely on a combination of linguistic knowledge and statistical association measures. Grammatical patterns, such as adjective-noun or noun-noun sequences are selected then ranked statistically, and the resulting ranked list is either used directly or submitted for manual filtering.

The linguistic filters used in typical term extraction systems have no obvious connection with the criteria that linguists would argue define a phrasal term (noncompositionality, fixed order, nonsubstitutability, etc.). They function, instead, to reduce the number of a priori improbable terms and thus improve precision. The association measure does the actual work of distinguishing between terms and plausible nonterms. A variety of methods have been applied, ranging from simple frequency (Justeson & Katz 1995), modified frequency measures such as c-values (Frantzi, Anadiou & Mima 2000, Maynard & Anadiou 2000) and standard statistical significance tests such as the t-test, the chi-squared test, and log-likelihood (Church and Hanks 1990, Dunning 1993), and information-based methods, e.g. pointwise mutual information (Church & Hanks 1990).

Several studies of the performance of lexical association metrics suggest significant room for improvement, but also variability among tasks.

One series of studies (Krenn 1998, 2000; Evert & Krenn 2001, Krenn & Evert 2001; also see Evert 2004) focused on the use of association metrics to identify the best candidates in particular grammatical constructions, such as adjective-noun pairs or verb plus prepositional phrase constructions, and compared the performance of simple frequency to several common measures (the log-likelihood, the t-test, the chi-squared test, the dice coefficient, relative entropy and mutual information). In Krenn & Evert 2001, frequency outperformed mutual information though not the t-test, while in Evert and Krenn 2001, log-likelihood and the t-test gave the best results, and mutual information again performed worse than frequency. However, in all these studies performance was generally low, with precision falling rapidly after the very highest ranked phrases in the list.

By contrast, Schone and Jurafsky (2001) evaluate the identification of phrasal terms without grammatical filtering on a 6.7 million word extract from the TREC databases, applying both WordNet and online dictionaries as gold standards. Once again, the general level of performance was low, with precision falling off rapidly as larger portions of the n-best list were included, but they report better performance with statistical and information theoretic measures (including mutual information) than with frequency. The overall pattern appears to be one where lexical association measures in general have very low precision and recall on unfiltered data, but perform far better when combined with other features which select linguistic patterns likely to function as phrasal terms.

The relatively low precision of lexical association measures on unfiltered data no doubt has multiple explanations, but a logical candidate is the failure or inappropriacy of underlying statistical assumptions. For instance, many of the tests assume a normal distribution, despite the highly skewed nature of natural language frequency distributions, though this is not the most important consideration except at very low *n* (cf. Moore 2004, Evert 2004, ch. 4). More importantly, statistical and information-based metrics such as the log-likelihood and mutual information measure significance or informativeness relative to the assumption that the selection of component terms is statistically independent. But of course the possibilities for combinations of words are anything but random and independent. Use of linguistic filters such as "attributive adjective followed by noun" or "verb plus modifying prepositional phrase" arguably has the effect of selecting a subset of the language for which the standard null hypothesis -- that any word may freely be combined with any other word -- may be much more accurate. Additionally, many of the association measures are defined only for bigrams, and do not generalize well to phrasal terms of varying length.

The purpose of this paper is to explore whether the identification of candidate phrasal terms can be improved by adopting a heuristic which seeks to take certain of these statistical issues into account. The method to be presented here, the *mutual rank ratio*, is a nonparametric rank-based approach which appears to perform significantly better than the standard association metrics.

The body of the paper is organized as follows: Section 2 will introduce the statistical considerations which provide a rationale for the mutual rank ratio heuristic and outline how it is calculated. Section 3 will present the data sources and evaluation methodologies applied in the rest of the paper. Section 4 will evaluate the mutual rank ratio statistic and several other lexical association measures on a larger corpus than has been used in previous evaluations. As will be shown below, the mutual rank ratio statistic recognizes phrasal terms more effectively than standard statistical measures.

2 Statistical considerations

2.1 Highly skewed distributions

As first observed e.g. by Zipf (1935, 1949) the frequency of words and other linguistic units tend to follow highly skewed distributions in which there are a large number of rare events. Zipf's formulation of this relationship for single word frequency distributions (Zipf's first law) postulates that the frequency of a word is inversely proportional to its rank in the frequency distribution, or more generally if we rank words by frequency and assign rank *z*, where the function $f_z(z,N)$ gives the frequency of rank z for a sample of size N, Zipf's first law states that:

$$f_z(z,N) = \frac{C}{z^{\alpha}}$$

where C is a normalizing constant and α is a free parameter that determines the exact degree of skew; typically with single word frequency data, α approximates 1 (Baayen 2001: 14). Ideally, an association metric would be designed to maximize its statistical validity with respect to the distribution which underlies natural language text -- which is if not a pure Zipfian distribution at least an LNRE (large number of rare events, cf. Baayen 2001) distribution with a very long tail, containing events which differ in probability by many orders of magnitude. Unfortunately, research on LNRE distributions focuses primarily on unigram distributions, and generalizations to bigram and n-gram distributions on large corpora are not as yet clearly feasible (Baayen 2001:221). Yet many of the best-performing lexical association measures, such as the t-test, assume normal distributions, (cf. Dunning 1993) or else (as with mutual information) eschew significance testing in favor of a generic information-theoretic approach. Various strategies could be adopted in this situation: finding a better model of the distribution,or adopting a nonparametric method.

2.2 The independence assumption

Even more importantly, many of the standard lexical association measures measure significance (or information content) against the default assumption that word-choices are statistically independent events. This assumption is built into the highest-performing measures as observed in Evert & Krenn 2001, Krenn & Evert 2001 and Schone & Jurafsky 2001.

This is of course untrue, and justifiable only as a simplifying idealization in the absence of a better model. The actual probability of any sequence of words is strongly influenced by the base grammatical and semantic structure of language, particularly since phrasal terms usually conform to the normal rules of linguistic structure. What makes a compound noun, or a verb-particle construction, into a phrasal term is not deviation from the base grammatical pattern for noun-noun or verb-particle structures, but rather a further pattern (of meaning and usage and thus heightened frequency) superimposed on the normal linguistic base. There are, of course, entirely aberrant phrasal terms, but they constitute the exception rather than the rule.

This state of affairs poses something of a chicken-and-the-egg problem, in that statistical parsing models have to estimate probabilities from the same base data as the lexical association measures, so the usual heuristic solution as noted above is to impose a linguistic filter on the data, with the association measures being applied only to the subset thus selected. The result is in effect a constrained statistical model in which the independence assumption is much more accurate. For instance, if the universe of statistical possibilities is restricted to the set of sequences in which an adjective is followed by a noun, the null hypothesis that word choice is independent -- i.e., that any adjective may precede any noun -- is a reasonable idealization. Without filtering, the independence assumption yields the much less plausible null hypothesis that any word may appear in any order.

It is thus worth considering whether there are any ways to bring additional information to bear on the problem of recognizing phrasal terms without presupposing statistical independence.

2.3 Variable length; alternative/overlapping phrases

Phrasal terms vary in length. Typically they range from about two to six words in length, but critically we cannot judge whether a phrase is lexical without considering both shorter and longer sequences.

That is, the statistical comparison that needs to be made must apply in principle to the entire set of word sequences that must be distinguished from phrasal terms, including longer sequences, subsequences, and overlapping sequences, despite the fact that these are not statistically independent events. Of the association metrics mentioned thus far, only the C-Value method attempts to take direct notice of such word sequence information, and then only as a modification to the basic information provided by frequency.

Any solution to the problem of variable length must enable normalization allowing direct comparison of phrases of different length. Ideally, the solution would also address the other issues -- the independence assumption and the skewed distributions typical of natural language data.

2.4 Mutual expectation

An interesting proposal which seeks to overcome the variable-length issue is the *mutual expectation* metric presented in Dias, Guilloré, and Lopes (1999) and implemented in the SENTA system (Gil and Dias 2003a). In their approach, the frequency of a phrase is normalized by taking into account the relative probability of each word compared to the phrase.

Dias, Guilloré, and Lopes take as the foundation of their approach the idea that the cohesiveness of a text unit can be measured by measuring how strongly it resists the loss of any component term. This is implemented by considering, for any n-gram, the set of [continuous or discontinuous] (n-1)-grams which can be formed by deleting one word from the n-gram. A *normalized expectation* for the n-gram is then calculated as follows:

$$\frac{p([w_1, w_2 ... w_n])}{FPE([w_1, w_2 ... w_n])}$$

where [w_1, w_2 ... w_n] is the phrase being evaluated and FPE([w_1, w_2 ... w_n]) is:

$$\frac{1}{n}\left(p([w_1, w_2 ... w_n]) + \sum_{i=1}^{n} p\left([w_1 ... \overset{\wedge}{w_i} .... w_n] \right) \right)$$

where w_i is the term omitted from the n-gram.

They then calculate mutual expectation as the product of the probability of the n-gram and its normalized expectation.

This statistic is of interest for two reasons: first, it provides a single statistic that can be applied to n-grams of any length; second, it is not based upon the independence assumption. The core statistic, normalized expectation, is essentially frequency with a penalty if a phrase contains component parts significantly more frequent than the phrase itself.

It is of course an empirical question how well mutual expectation performs (and we shall examine this below) but mutual expectation is not in any sense a significance test. That is, if we are examining a phrase like the *east end*, the conditional probability of *east* given *[__ end]* or of *end* given *[__ east]* may be relatively low (since other words can appear in that context) and yet the phrase might still be very lexicalized if the association of both words with this context were significantly stronger than their association for other phrases. That is, to the extent that phrasal terms follow the regular patterns of the language, a phrase might have a relatively low conditional probability (given the wide range of alternative phrases following the same basic linguistic patterns) and thus have a low mutual expectation yet still occur far more often than one would expect from chance.

In short, the fundamental insight -- assessing how tightly each word is bound to a phrase -- is worth adopting. There is, however, good reason to suspect that one could improve on this method by assessing relative statistical significance for each component word without making the independence assumption. In the heuristic to be outlined below, a nonparametric method is proposed. This method is novel: not a modification of mutual expectation, but a new technique based on ranks in a Zipfian frequency distribution.

2.5 Rank ratios and mutual rank ratios

This technique can be justified as follows. For each component word in the n-gram, we want to know whether the n-gram is more probable for that word than we would expect given its behavior with other words. Since we do not know what the expected shape of this distribution is going to be, a nonparametric method using ranks is in order, and there is some reason to think that frequency rank regardless of n-gram size will be useful. In particular, Ha, Sicilia-Garcia, Ming and Smith (2002) show that Zipf's law can be extended to the combined frequency distribution of n-grams of varying length up to rank 6, which entails that the relative rank of words in such a combined distribution provide a useful estimate of relative probability. The availability of new techniques for handling large sets of n-gram data (e.g. Gil & Dias 2003b) make this a relatively feasible task.

Thus, given a phrase like *east end*, we can rank how often __ *end* appears with *east* in comparison to how often other phrases appear with *east*.That is, if *{__ end, __side, the __, toward the __, etc.}* is the set of (variable length) n-gram contexts associated with *east* (up to a length cutoff), then the ***actual rank*** of __ *end* is the rank we calculate by ordering all contexts by the frequency with which the actual word appears in the context.

We also rank the set of contexts associated with *east* by their overall corpus frequency. The resulting ranking is the ***expected rank*** of __ *end* based upon how often the competing contexts appear regardless of which word fills the context.

The rank ratio (RR) for the word given the context can then be defined as:

$$\text{RR(word,context)} = \frac{ER(word, context)}{AR(word, context)}$$

where ER is the expected rank and AR is the actual rank. A normalized, or mutual rank ratio for the n-gram can then be defined as

$$\sqrt[n]{RR(w_{1,[\_\_\ w2....wn]}) * RR(w_{2,[w1\ \_\_\ ...wn]})... * RR(w_{n,[w1,\ w2...\ \_]})}$$

The motivation for this method is that it attempts to address each of the major issues outlined above by providing a nonparametric metric which does not make the independence assumption and allows scores to be compared across n-grams of different lengths.

A few notes about the details of the method are in order. Actual ranks are assigned by listing all the contexts associated with each word in the corpus, and then ranking contexts by word, assigning the most frequent context for word n the rank 1, next next most frequent rank 2, etc. Tied ranks are given the median value for the ranks occupied by the tie, e.g., if two contexts with the same frequency would occupy ranks 2 and 3, they are both assigned rank 2.5. Expected ranks are calculated for the same set of contexts using the same algorithm, but substituting the unconditional frequency of the (n-1)-gram for the gram's frequency with the target word.[1]

3 Data sources and methodology

The Lexile Corpus is a collection of documents covering a wide range of reading materials such as a child might encounter at school, more or less evenly divided by Lexile (reading level) rating to cover all levels of textual complexity from kindergarten to college. It contains in excess of 400 million words of running text, and has been made available to the Educational Testing Service under a research license by Metametrics Corporation.

This corpus was tokenized using an in-house tokenization program, *toksent*, which treats most punctuation marks as separate tokens but makes single tokens out of common abbreviations, numbers like *1,500*, and words like *o'clock*. It should be noted that some of the association measures are known to perform poorly if punctuation marks and common stopwords are

[1] In this study the rank-ratio method was tested for bigrams and trigrams only, due to the small number of WordNet gold standard items greater than two words in length. Work in progress will assess the metrics' performance on n-grams of orders four through six.

included; therefore, n-gram sequences containing punctuation marks and the 160 most frequent word forms were excluded from the analysis so as not to bias the results against them. Separate lists of bigrams and trigrams were extracted and ranked according to several standard word association metrics. Rank ratios were calculated from a comparison set consisting of all contexts derived by this method from bigrams and trigrams, e.g., contexts of the *form word1__*, *___word2*, *___word1 word2*, *word1 ___ word3*, and *word1 word2 ___*.[2]

Table 1 lists the standard lexical association measures tested in section four[3].

The logical evaluation method for phrasal term identification is to rank n-grams using each metric and then compare the results against a gold standard containing known phrasal terms. Since Schone and Jurafsky (2001) demonstrated similar results whether WordNet or online dictionaries were used as a gold standard, WordNet was selected. Two separate lists were derived containing two- and three-word phrases. The choice of WordNet as a gold standard tests ability to predict general dictionary headwords rather than technical terms, appropriate since the source corpus consists of nontechnical text.

Following Schone & Jurafsky (2001), the bigram and trigram lists were ranked by each statistic then scored against the gold standard, with results evaluated using a figure of merit (FOM) roughly characterizable as the area under the precision-recall curve. The formula is:

$$\frac{1}{K}\sum_{i=1}^{k} P_i$$

where P_i (precision at *i*) equals i/H_i, and H_i is the number of n-grams into the ranked n-gram list required to find the i^{th} correct phrasal term.

It should be noted, however, that one of the most pressing issues with respect to phrasal terms is that they display the same skewed, long-tail distribution as ordinary words, with a large proportion of the total displaying very low frequencies. This can be measured by considering

METRIC	FORMULA
Frequency (Guiliano, 1964)	f_{xy}
Pointwise Mutual Information [PMI] (Church & Hanks, 1990)	$\log_2\left(P_{xy} / P_x P_y\right)$
True Mutual Information [TMI] (Manning, 1999)	$P_{xy}\log_2\left(P_{xy} / P_x P_y\right)$
Chi-Squared (χ^2) (Church and Gale, 1991)	$\sum_{\substack{i\in\{X,\overline{X}\}\\ j\in\{Y,\overline{Y}\}}} \frac{(f_{ij}-\zeta_{ij})^2}{\zeta_{ij}}$
T-Score (Church & Hanks, 1990)	$\frac{x_1 - x_2}{\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}}$
C-Values[4] (Frantzi, Anadiou & Mima 2000)	$\left\{\begin{array}{l}\log_2\lvert\alpha\rvert\cdot f(\alpha)_{\alpha \text{ is not nested}}\\ \log_2\lvert\alpha\rvert\cdot f(\alpha)\\ \qquad -\frac{1}{P(T_a)}\sum_{b\in T_a} f(b)\end{array}\right\}$ where α is the candidate string f(α) is its frequency in the corpus Tα is the set of candidate terms that contain α P(Tα) is the number of these candidate terms

Table 1. Some Lexical Association Measures

the overlap between WordNet and the Lexile corpus. A list of 53,764 two-word phrases were extracted from WordNet, and 7,613 three-word phrases. Even though the Lexile corpus is quite large -- in excess of 400 million words of running text -- only 19,939 of the two-word phrases and

[2] Excluding the 160 most frequent words prevented evaluation of a subset of phrasal terms such as verbal idioms like *act up* or *go on*. Experiments with smaller corpora during preliminary work indicated that this exclusion did not appear to bias the results.

[3] Schone & Jurafsky's results indicate similar results for log-likelihood & T-score, and strong parallelism among information-theoretic measures such as Chi-Squared, Selectional Association (Resnik 1996), Symmetric Conditional Probability (Ferreira and Pereira Lopes, 1999) and the Z-Score (Smadja 1993). Thus it was not judged necessary to replicate results for all methods covered in Schone & Jurafsky (2001).

[4] Due to the computational cost of calculating C-Values over a very large corpus, C-Values were calculated over bigrams and trigrams only. More sophisticated versions of the C-Value method such as NC-values were not included as these incorporate linguistic knowledge and thus fall outside the scope of the study.

1,700 of the three-word phrases are attested in the Lexile corpus. 14,045 of the 19,939 attested two-word phrases occur at least 5 times, 11,384 occur at least 10 times, and only 5,366 occur at least 50 times; in short, the strategy of cutting off the data at a threshold sacrifices a large percent of total recall. Thus one of the issues that needs to be addressed is the accuracy with which lexical association measures can be extended to deal with relatively sparse data, e.g., phrases that appear less than ten times in the source corpus.

A second question of interest is the effect of filtering for particular linguistic patterns. This is another method of prescreening the source data which can improve precision but damage recall. In the evaluation bigrams were classified as N-N and A-N sequences using a dictionary template, with the expected effect. For instance, if the WordNet two word phrase list is limited only to those which could be interpreted as noun-noun or adjective noun sequences, N>=5, the total set of WordNet terms that can be retrieved is reduced to 9,757..

4 Evaluation

Schone and Jurafsky's (2001) study examined the performance of various association metrics on a corpus of 6.7 million words with a cutoff of N=10. The resulting n-gram set had a maximum recall of 2,610 phrasal terms from the WordNet gold standard, and found the best figure of merit for any of the association metrics even with linguistic filterering to be 0.265. On the significantly larger Lexile corpus N must be set higher (around N=50) to make the results comparable. The statistics were also calculated for N=50, N=10 and N=5 in order to see what the effect of including more (relatively rare) n-grams would be on the overall performance for each statistic. Since many of the statistics are defined without interpolation only for bigrams, and the number of WordNet trigrams at N=50 is very small, the full set of scores were only calculated on the bigram data. For trigrams, in addition to rank ratio and frequency scores, extended pointwise mutual information and true mutual information scores were calculated using the formulas log $(P_{xyz}/P_xP_y\ P_z)$) and P_{xyz} log $(P_{xyz}/P_xP_y\ P_z)$). Also, since the standard lexical association metrics cannot be calculated across different n-gram types, results for bigrams and trigrams are presented separately for purposes of comparison.

The results are are shown in Tables 2-5. Two points should should be noted in particular. First, the rank ratio statistic outperformed the other association measures tested across the board. Its best performance, a score of 0.323 in the part of speech filtered condition with N=50, outdistanced

METRIC	POS Filtered	Unfiltered
RankRatio	0.323	0.196
Mutual Expectancy	0.144	0.069
TMI	0.209	0.096
PMI	0.287	0.166
Chi-sqr	0.285	0.152
T-Score	0.154	0.046
C-Values	0.065	0.048
Frequency	0.130	0.044

Table 2. Bigram Scores for Lexical Association Measures with N=50

METRIC	POS Filtered	Unfiltered
RankRatio	0.218	0.125
MutualExpectation	0.140	0.071
TMI	0.150	0.070
PMI	0.147	0.065
Chi-sqr	0.145	0.065
T-Score	0.112	0.048
C-Values	0.096	0.036
Frequency	0.093	0.034

Table 3. Bigram Scores for Lexical Association Measures with N=10

METRIC	POS Filtered	Unfiltered
RankRatio	0.188	0.110
Mutual Expectancy	0.141	0.073
TMI	0.131	0.063
PMI	0.108	0.047
Chi-sqr	0.107	0.047
T-Score	0.098	0.043
C-Values	0.084	0.031
Frequency	0.081	0.021

Table 4. Bigram Scores for Lexical Association Measures with N=5

METRIC	N=50	N=10	N=5
RankRatio	0.273	0.137	0.103
PMI	0.219	0.121	0.059
TMI	0.137	0.074	0.056
Frequency	0.089	0.047	0.035

Table 5. Trigram scores for Lexical Association Measures at N=50, 10 and 5 without linguistic filtering.

the best score in Schone & Jurafsky's study (0.265), and when large numbers of rare bigrams were included, at N=10 and N=5, it continued to outperform the other measures. Second, the results were generally consistent with those reported in the literature, and confirmed Schone & Jurafsky's observation that the information-theoretic measures (such as mutual information and chi-squared) outperform frequency-based measures (such as the T-score and raw frequency.)[5]

4.1 Discussion

One of the potential strengths of this method is that is allows for a comparison between n-grams of varying lengths. The distribution of scores for the gold standard bigrams and trigrams appears to bear out the hypothesis that the numbers are comparable across n-gram length. Trigrams constitute approximately four percent of the gold standard test set, and appear in roughly the same percentage across the rankings; for instance, they consistute 3.8% of the top 10,000 ngrams ranked by mutual rank ratio. Comparison of trigrams with their component bigrams also seems consistent with this hypothesis; e.g., the bigram *Booker T.* has a higher mutual rank ratio than the trigram *Booker T. Washington*, which has a higher rank that the bigram *T. Washington*. These results suggest that it would be worthwhile to examine how well the method succeeds at ranking n-grams of varying lengths, though the limitations of the current evaluation set to bigrams and trigrams prevented a full evaluation of its effectiveness across n-grams of varying length.

The results of this study appear to support the conclusion that the Mutual Rank Ratio performs notably better than other association measures on this task. The performance is superior to the next-best measure when N is set as low as 5 (0.110 compared to 0.073 for Mutual Expectation and 0.063 for true mutual information and less than .05 for all other metrics). While this score is still fairly low, it indicates that the measure performs relatively well even when large numbers of low-probability n-grams are included. An examination of the n-best list for the Mutual Rank ratio at N=5 supports this contention.

The top 10 bigrams are:

Julius Caesar, Winston Churchill, potato chips, peanut butter, Frederick Douglass, Ronald Reagan, Tia Dolores, Don Quixote, cash register, Santa Claus

At ranks 3,000 to 3,010, the bigrams are:

Ted Williams, surgical technicians, Buffalo Bill, drug dealer, Lise Meitner, Butch Cassidy, Sandra Cisneros, Trey Granger, senior prom, Ruta Skadi

At ranks 10,000 to 10,010, the bigrams are:

egg beater, sperm cells, lowercase letters, methane gas, white settlers, training program, instantly recognizable, dried beef, television screens, vienna sausages

In short, the n-best list returned by the mutual rank ratio statistic appears to consist primarily of phrasal terms far down the list, even when N is as low as 5. False positives are typically: (i) morphological variants of established phrases; (ii) bigrams that are part of longer phrases, such as *cream sundae* (from *ice cream sundae*); (iii) examples of highly productive constructions such as *an artist*, *three categories* or *January 2.*

The results for trigrams are relatively sparse and thus less conclusive, but are consistent with the bigram results: the mutual rank ratio measure performs best, with top ranking elements consistently being phrasal terms.

Comparison with the n-best list for other metrics bears out the qualitative impression that the rank ratio is performing better at selecting phrasal terms even without filtering. The top ten bigrams for the true mutual information metric at N=5 are:

a little, did not, this is, united states, new york, know what, a good, a long, a moment, a small

Ranks 3000 to 3010 are:

waste time, heavily on, earlier than, daddy said, ethnic groups, tropical rain, felt sure, raw materials, gold medals, gold rush

Ranks 10,000 to 10,010 are:

quite close, upstairs window, object is, lord god, private schools, nat turner, fire going, bering sea,little higher, got lots

The behavior is consistent with known weaknesses of true mutual information -- its tendency to overvalue frequent forms.

Next, consider the n-best lists for log-likelihood at N=5. The top ten n-grams are:

sheriff poulson, simon huggett, robin redbreast, eric torrosian, colonel hillandale, colonel sapp, nurse leatheran, st. catherines, karen torrio, jenny yonge

N-grams 3000 to 3010 are:

comes then, stuff who, dinner get, captain see, tom see, couple get, fish see, picture go, building go, makes will, pointed way

[5] Schone and Jurafsky's results differ from Krenn & Evert (2001)'s results, which indicated that frequency performed better than the statistical measures in almost every case. However, Krenn and Evert's data consisted of n-grams preselected to fit particular collocational patterns. Frequency-based metrics seem to be particularly benefited by linguistic prefiltering.

N-grams 10000 to 10010 are:

sayings is, writ this, llama on, undoing this, dwahro did, reno on, squirted on, hardens like, mora did, millicent is, vets did

Comparison thus seems to suggest that if anything the quality of the mutual rank ratio results are being understated by the evaluation metric, as the metric is returning a large number of phrasal terms in the higher portion of the n-best list that are absent from the gold standard.

Conclusion

This study has proposed a new method for measuring strength of lexical association for candidate phrasal terms based upon the use of Zipfian ranks over a frequency distribution combining n-grams of varying length. The method is related in general philosophy of Mutual Expectation, in that it assesses the strenght of connection for each word to the combined phrase; it differs by adopting a nonparametric measure of strength of association. Evaluation indicates that this method may outperform standard lexical association measures, including mutual information, chi-squared, log-likelihood, and the T-score.

References

Baayen, R. H. (2001) *Word Frequency Distributions*. Kluwer: Dordrecht.

Boguraev, B. and C. Kennedy (1999). Applications of Term Identification Technology: Domain Description and Content Characterization. *Natural Language Engineering* 5(1):17-44.

Choueka, Y. (1988). Looking for needles in a haystack or locating interesting collocation expressions in large textual databases. *Proceedings of the RIAO*, pages 38-43.

Church, K.W., and P. Hanks (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics 16(1):*22-29.

Dagan, I. and K.W. Church (1994). Termight: Identifying and translating technical terminology**.** *ACM International Conference Proceeding Series: Proceedings of the fourth conference on Applied natural language processing***,** pages 39-40.

Daille, B. 1996. "Study and Implementation of Combined Techniques from Automatic Extraction of Terminology". Chap. 3 of "The Balancing Act": *Combining Symbolic and Statistical Approaches to Kanguage* (Klavans, J., Resnik, P. (eds.)), pages 49-66.

Dias, G., S. Guilloré, and J.G. Pereira Lopes (1999), Language independent automatic acquisition of rigid multiword units from unrestricted text corpora. *TALN*, p. 333-338.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics 19(1):* 65-74.

Evert, S. (2004). The Statistics of Word Cooccurrences: Word Pairs and Collocations. Phd Thesis, Institut für maschinelle Sprachverarbeitung, University of Stuttgart.

Evert, S. and B. Krenn. (2001). Methods for the Qualitative Evaluation of Lexical Association Measures. Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, pages 188-195.

Ferreira da Silva, J. and G. Pereira Lopes (1999). A local maxima method and a fair dispersion normalization for extracting multiword units from corpora. *Sixth Meeting on Mathematics of Language*, pages 369-381.

Frantzi, K., S. Ananiadou, and H. Mima. (2000). Automatic recognition of multiword terms: the C-Value and NC-Value Method. *International Journal on Digital Libraries* 3(2):115-130.

Gil, A. and G. Dias. (2003a). Efficient Mining of Textual Associations. *International Conference on Natural Language Processing and Knowledge Engineering.* Chengqing Zong (eds.) pages 26-29.

Gil, A. and G. Dias (2003b). Using masks, suffix array-based data structures, and multidimensional arrays to compute positional n-gram statistics from corpora. In Proceedings of the Workshop on Multiword Expressions of the 41st Annual Meeting of the Association of Computational Linguistics, pages 25-33.

Ha, L.Q., E.I. Sicilia-Garcia, J. Ming and F.J. Smith. (2002), "Extension of Zipf's law to words and phrases", *Proceedings of the 19th International Conference on Computational Linguistics (COLING'2002),* pages 315-320.

Jacquemin, C. and E. Tzoukermann. (1999). NLP for Term Variant Extraction: Synergy between Morphology, Lexicon, and Syntax. *Natural Language Processing Information Retrieval*, pages 25-74. Kuwer, Boston, MA, U.S.A.

Jacquemin, C., J.L. Klavans and E. Tzoukermann (1997). Expansion of multiword terms for indexing and retrieval using morphology and syntax. Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, pages 24-31.

Johansson, C. 1994b, Catching the Cheshire Cat, In *Proceedings of COLING 94*, Vol. II, pages 1021 - 1025.

Johansson, C. 1996. Good Bigrams. In Proceedings from the 16th International Conference on Computational Linguistics (COLING-96), pages 592-597.

Justeson, J.S. and S.M. Katz (1995). Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering* 1:9-27.

Krenn, B. 1998. Acquisition of Phraseological Units from Linguistically Interpreted Corpora. A Case Study on German PP-Verb Collocations. Proceedings of ISP-98, pages 359-371.

Krenn, B. 2000. Empirical Implications on Lexical Association Measures. Proceedings of The Ninth EURALEX International Congress.

Krenn, B. and S. Evert. 2001. Can we do better than frequency? A case study on extracting PP-verb collocations. Proceedings of the ACL Workshop on Collocations, pages 39-46.

Lin, D. 1998. Extracting Collocations from Text Corpora. *First Workshop on Computational Terminology*, pages 57-63

Lin, D. 1999. Automatic Identification of Non-compositional Phrases, In *Proceedings of The 37th Annual Meeting of the Association For Computational Lingusitics*, pages 317-324.

Manning, C.D. and H. Schütze. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, U.S.A.

Maynard, D. and S. Ananiadou. (2000). Identifying Terms by their Family and Friends. COLING 2000, pages 530-536.

Pantel, P. and D. Lin. (2001). A Statistical Corpus-Based Term Extractor. In: Stroulia, E. and Matwin, S. (Eds.) *AI 2001, Lecture Notes in Artificial Intelligence*, pages 36-46. Springer-Verlag.

Resnik, P. (1996). Selectional constraints: an information-theoretic model and its computational realization. *Cognition 61*: 127-159.

Schone, P. and D. Jurafsky, 2001. Is Knowledge-Free Induction of Multiword Unit Dictionary Headwords a Solved Problem? Proceedings of *Empirical Methods in Natural Language Processing*, pages 100-108.

Sekine, S., J. J. Carroll, S. Ananiadou, and J. Tsujii. 1992. *Automatic Learning for Semantic Collocation.* Proceedings of the 3rd Conference on Applied Natural Language Processing, pages 104-110.

Shimohata, S., T. Sugio, and J. Nagata. (1997). Retrieving collocations by co-occurrences and word order constraints. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 476-481.

Smadja, F. (1993). Retrieving collocations from text: Xtract. Computational Linguistics, 19:143-177.

Thanapoulos, A., N. Fakotakis and G. Kokkinkais. 2002. Comparaitve Evaluation of Collocation Extraction Metrics. *Proceedings of the LREC 2002 Conference*, pages 609-613.

Zipf, P. (1935). *Psychobiology of Language.* Houghton-Mifflin, New York, New York.

Zipf, P.(1949). *Human Behavior and the Principle of Least Effort.* Addison-Wesley, Cambridge, Mass.

Automatic Acquisition of Adjectival Subcategorization from Corpora

Jeremy Yallop[*], **Anna Korhonen, and Ted Briscoe**
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 OFD, UK
yallop@cantab.net, {Anna.Korhonen, Ted.Briscoe}@cl.cam.ac.uk

Abstract

This paper describes a novel system for acquiring adjectival subcategorization frames (SCFs) and associated frequency information from English corpus data. The system incorporates a decision-tree classifier for 30 SCF types which tests for the presence of grammatical relations (GRs) in the output of a robust statistical parser. It uses a powerful pattern-matching language to classify GRs into frames hierarchically in a way that mirrors inheritance-based lexica. The experiments show that the system is able to detect SCF types with 70% precision and 66% recall rate. A new tool for linguistic annotation of SCFs in corpus data is also introduced which can considerably alleviate the process of obtaining training and test data for subcategorization acquisition.

1 Introduction

Research into automatic acquisition of lexical information from large repositories of unannotated text (such as the web, corpora of published text, etc.) is starting to produce large scale lexical resources which include frequency and usage information tuned to genres and sublanguages. Such resources are critical for natural language processing (NLP), both for enhancing the performance of state-of-art statistical systems and for improving the portability of these systems between domains.

One type of lexical information with particular importance for NLP is subcategorization. Access to an accurate and comprehensive subcategorization lexicon is vital for the development of successful parsing technology (e.g. (Carroll et al., 1998b), important for many NLP tasks (e.g. automatic verb classification (Schulte im Walde and Brew, 2002)) and useful for any application which can benefit from information about predicate-argument structure (e.g. Information Extraction (IE) (Surdeanu et al., 2003)).

The first systems capable of automatically learning a small number of verbal subcategorization frames (SCFs) from English corpora emerged over a decade ago (Brent, 1991; Manning, 1993). Subsequent research has yielded systems for English (Carroll and Rooth, 1998; Briscoe and Carroll, 1997; Korhonen, 2002) capable of detecting comprehensive sets of SCFs with promising accuracy and demonstrated success in application tasks (e.g. (Carroll et al., 1998b; Korhonen et al., 2003)), besides systems for a number of other languages (e.g. (Kawahara and Kurohashi, 2002; Ferrer, 2004)).

While there has been considerable research into acquisition of verb subcategorization, we are not aware of any systems built for adjectives. Although adjectives are syntactically less multivalent than verbs, and although verb subcategorization distribution data appears to offer the greatest potential boost in parser performance, accurate and comprehensive knowledge of the many adjective SCFs can improve the accuracy of parsing at several levels

[*]Part of this research was conducted while this author was at the University of Edinburgh Laboratory for Foundations of Computer Science.

(from tagging to syntactic and semantic analysis). Automatic SCF acquisition techniques are particularly important for adjectives because extant syntax dictionaries provide very limited coverage of adjective subcategorization.

In this paper we propose a method for automatic acquisition of adjectival SCFs from English corpus data. Our method has been implemented using a decision-tree classifier which tests for the presence of grammatical relations (GRs) in the output of the RASP (Robust Accurate Statistical Parsing) system (Briscoe and Carroll, 2002). It uses a powerful task-specific pattern-matching language which enables the frames to be classified hierarchically in a way that mirrors inheritance-based lexica. As reported later, the system is capable of detecting 30 SCFs with an accuracy comparable to that of best state-of-art verbal SCF acquisition systems (e.g. (Korhonen, 2002)).

Additionally, we present a novel tool for linguistic annotation of SCFs in corpus data aimed at alleviating the process of obtaining training and test data for subcategorization acquisition. The tool incorporates an intuitive interface with the ability to significantly reduce the number of frames presented to the user for each sentence.

We discuss adjectival subcategorization in section 2 and introduce the system for SCF acquisition in section 3. Details of the annotation tool and the experimental evaluation are supplied in section 4. Section 5 provides discussion on our results and future work, and section 6 summarises the paper.

2 Adjectival Subcategorization

Although the number of SCF types for adjectives is smaller than the number reported for verbs (e.g. (Briscoe and Carroll, 1997)), adjectives nevertheless exhibit rich syntactic behaviour. Besides the common attributive and predicative positions there are at least six further positions in which adjectives commonly occur (see figure 1). Adjectives in predicative position can be further classified according to the nature of the arguments with which they combine — finite and non-finite clauses and noun phrases, phrases with and without complementisers, etc. — and whether they occur as subject or object. Additional distinctions can be made concerning such features as the mood of the complement (mandative, interrogative, etc.), preferences for particular prepositions and whether the subject is extraposed.

Attributive	“The young man”
Predicative	“He is young”
Postpositive	“Anyone [who is] young can do it”
Predeterminer	“such a young man”; “so young a man”
Fused modifier-head	“the younger of them”; “the young”
Predicative adjunct	“he died young”
Supplementive clause	“Young, he was plain in appearance”
Contingent clause	“When young, he was lonely”

Figure 1: Fundamental adjectival frames

Even ignoring preposition preference, there are more than 30 distinguishable adjectival SCFs. Some fairly extensive frame sets can be found in large syntax dictionaries, such as COMLEX (31 SCFs) (Wolff et al., 1998) and ANLT (24 SCFs) (Boguraev et al., 1987). While such resources are generally accurate, they are disappointingly incomplete: none of the proposed frame sets in the well-known resources subsumes the others, the coverage of SCF types for individual adjectives is low, and (accurate) information on the relative frequency of SCFs for each adjective is absent.

The inadequacy of manually-created dictionaries and the difficulty of adequately enhancing and maintaining the information by hand was a central motivation for early research into automatic subcategorization acquisition. The focus heretofore has remained firmly on verb subcategorization, but this is not sufficient, as countless examples show. Knowledge of adjectival subcategorization can yield further improvements in tagging (e.g. distinguishing between “to” as an infinitive marker and as a true preposition), parsing (e.g. distinguishing between PP-arguments and adjuncts), and semantic analysis. For example, if John is both *easy* and *eager* to please then *we* know that he is the recipient of pleasure in the first instance and desirous of providing it in the second, but a computational system cannot determine this without knowledge of the subcategorization of the two adjectives. Likewise, a natural language generation system can legitimately apply the extraposition transformation to the first case, but not to the second: It is “easy to please John”, but not "eager" to do so, at least if "it" be expletive. Similar examples abound.

Many of the difficulties described in the literature on acquiring verb subcategorization also arise in the adjectival case. The most apparent is data sparsity: among the 100M-word British National Corpus (BNC) (Burnard, 1995), the RASP tools find 124,120 distinct adjectives, of which 70,246 occur only once, 106,464 fewer than ten times and 119,337 fewer than a hundred times. There are fewer than 1,000 adjectives in the corpus which have more than 1,000 occurrences. Both adjective and SCF frequencies have Zipfian distributions; consequently, even the largest corpora may contain only single instances of a particular adjective-SCF combination, which is generally insufficient for classification.

3 Description of the System

Besides focusing on adjectives, our approach to SCF acquisition differs from earlier work in a number of ways. A common strategy in existing systems (e.g. (Briscoe and Carroll, 1997)) is to extract SCFs from parse trees, introducing an unnecessary dependence on the details of a particular parser. In our approach the patterns are extracted from GRs — representations of head-complement relations which are designed to be largely parser-independent — making the techniques more widely applicable and allowing classification to operate at a higher level. Further, most existing systems work by classifying corpus occurrences into individual, mutually independent SCFs. We adopt instead a hierarchical approach, viewing frames that share features as descendants of a common parent frame. The benefits are severalfold: specifying each feature only once makes the system both more efficient and easier to understand and maintain, and the multiple inheritance hierarchy reflects the hierarchy of lexical types found in modern grammars where relationships between similar frames are represented explicitly[1].

Our acquisition process consists of two main steps: 1) extracting GRs from corpus data, and 2) feeding the GRs as input to the classifier which incrementally matches parts of the GR sets to decide which branches of a decision-tree to follow. The leaves of the tree correspond to SCFs. The details of these two steps are provided in the subsequent sections, respectively[2].

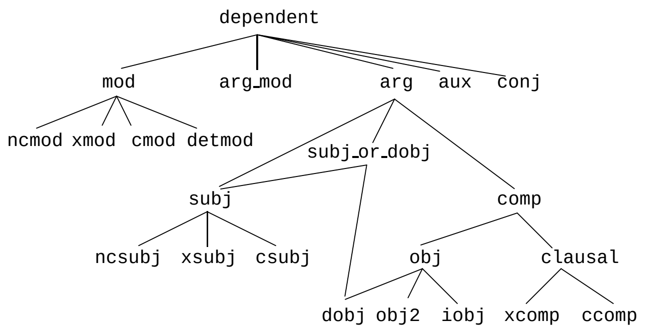


Figure 2: The GR hierarchy used by RASP

3.1 Obtaining Grammatical Relations

Attempts to acquire verb subcategorization have benefited from increasingly sophisticated parsers. We have made use of the RASP toolkit (Briscoe and Carroll, 2002) — a modular statistical parsing system which includes a tokenizer, tagger, lemmatiser, and a wide-coverage unification-based tag-sequence parser. The parser has several modes of operation; we invoked it in a mode in which GRs with associated probabilities are emitted even when a complete analysis of the sentence could not be found. In this mode there is wide coverage (over 98% of the BNC receives at least a partial analysis (Carroll and Briscoe, 2002)) which is useful in view of the infrequent occurrence of some of the SCFs, although combining the results of competing parses may in some cases result in an inconsistent or misleading combination of GRs.

The parser uses a scheme of GRs between lemmatised lexical heads (Carroll et al., 1998a; Briscoe et al., 2002). The relations are organized as a multiple-inheritance subsumption hierarchy where each subrelation extends the meaning, and perhaps the argument structure, of its parents (figure 2). For descriptions and examples of each relation, see (Carroll et al., 1998a).

The dependency relationships which the GRs embody correspond closely to the head-complement

[1]Compare the cogent argument for a inheritance-based lexicon in (Flickinger and Nerbonne, 1992), much of which can be applied unchanged to the taxonomy of SCFs.

[2]In contrast to almost all earlier work, there was no fi ltering stage involved in SCF acquisition. The classifi er was designed to operate with high precision, so fi ltering was less necessary.

$$\begin{bmatrix} \text{SUBJECT} & \text{NP}_{\boxed{1}}, \\ \text{ADJ-COMPS} & \left\langle \begin{bmatrix}\text{PVAL} & \text{“for”} \\ \text{NP} & \boxed{3}\end{bmatrix}_{\text{PP}}, \begin{bmatrix}\text{MOOD} & \text{to-infinitive} \\ \text{SUBJECT} & \boxed{3} \\ \text{OMISSION} & \boxed{1}\end{bmatrix}_{\text{VP}} \right\rangle \end{bmatrix}$$

Figure 3: Feature structure for SCF `adj-obj-for-to-inf`

```
(|These:1_DD2| |example+s:2_NN2| |of:3_IO|
 |animal:4_JJ| |senses:5_NN2| |be+:6_VBR|
 |relatively:7_RR| |easy:8_JJ| |for:9_IF|
 |we+:10_PPIO2| |to:11_TO| |comprehend:12_VV0|)
...
 xcomp(_               be+[6]       easy:[8])
  xmod(to[11]          be+[6]       comprehend:[12])
ncsubj(be+[6]          example+s[2] _)
 ncmod(for[9]          easy[8]      we+[10])
ncsubj(comprehend[12] we+[10], _)
...
```

Figure 4: GRs from RASP for `adj-obj-for-to-inf`

structure which subcategorization acquisition attempts to recover, which makes GRs ideal input to the SCF classifier. Consider the arguments of “easy” in the sentence:

> *These examples of animal senses are relatively easy for us to comprehend as they are not too far removed from our own experience.*

According to the COMLEX classification, this is an example of the frame `adj-obj-for-to-inf`, shown in figure 3, (using AVM notation in place of COMLEX s-expressions). Part of the output of RASP for this sentence (the full output includes 87 weighted GRs) is shown in figure 4[3].

Each instantiated GR in figure 4 corresponds to one or more parts of the feature structure in figure 3. `xcomp(_ be[6] easy[8])` establishes `be[6]` as the head of the `VP` in which `easy[8]` occurs as a complement. The first (`PP`)-complement is “for us”, as indicated by `ncmod(for[9] easy[8] we+[10])`, with “for” as `PFORM` and `we+` (“us”) as `NP`. The second complement is represented by `xmod(to[11] be+[6] comprehend[12])`: a *to-infinitive* `VP`. The `NP` headed by “examples” is marked as the subject of the frame by `ncsubj(be[6] examples[2])`, and `ncsubj(comprehend[12] we+[10])` corresponds to the coindexation marked by $\boxed{3}$: the subject of the `VP` is the `NP` of the `PP`. The only part of the feature structure which is not represented by the GRs is coindexation between the omitted direct object $\boxed{1}$ of the `VP`-complement and the subject of the whole clause.

[3]The format is slightly more complicated than that shown in (Carroll et al., 1998a): each argument that corresponds to a word consists of three parts: the lexeme, the part of speech tag, and the position (index) of the word in the sentence.

```
xcomp(_, [*;1;be-verb], ~)
xmod([to;*;to], 1, [*;2;vv0])
ncsubj(1, [*;3;noun/pronoun], _)
ncmod([for;*;if], ~, [*;4;noun/pronoun])
ncsubj(2, 4)
```

Figure 5: A pattern to match the frame `adj-obj-for-to-inf`

3.2 SCF Classifier

3.2.1 SCF Frames

We used for our classifier a modified version of the fairly extensive COMLEX frameset, including 30 SCFs. The COMLEX frameset includes mutually inconsistent frames, such as sentential complement with obligatory complementiser *that* and sentential complement with optional *that*. We modified the frameset so that an adjective can legitimately instantiate any combination of frames, which simplifies classification. We also added `simple-predicative` and `attributive` SCFs to the set, since these account for a substantial proportion of frame instances. Finally, frames which could only be distinguished by information not retained in the GRs scheme of the current version of the shallow parser were merged (e.g. the COMLEX frames `adj-subj-to-inf-rs` (“She was kind to invite me”) and `adj-to-inf` (“She was able to climb the mountain”)).

3.2.2 Classifier

The classifier operates by attempting to match the set of GRs associated with each sentence against various patterns. The patterns were developed by a combination of knowledge of the GRs and examining a set of training sentences to determine which relations were actually emitted by the parser for each SCF. The data used during development consisted of the sentences in the BNC in which one of the 23 adjectives[4] given as examples for SCFs in (Macleod

[4]The adjectives used for training were: *able, anxious, apparent, certain, convenient, curious, desirable, disappointed, easy, happy, helpful, imperative, impractical, insistent, kind, obvious, practical, preferable, probable, ridiculous, unaware, uncertain* and *unclear.*

et al., 1998) occur.

In our pattern matching language a pattern is a disjunction of sets of partially instantiated GRs with logic variables (*slots*) in place of indices, augmented by ordering constraints that restrict the possible instantiations of slots. A match is considered successful if the set of GRs can be unified with any of the disjuncts. Unification of a sentence-relation and a pattern-relation occurs when there is a one-to-one correspondence between sentence elements and pattern elements that includes a mapping from slots to indices (a *substitution*), and where atomic elements in corresponding positions share a common subtype.

Figure 5 shows a pattern for matching the SCF `adj-obj-for-to-inf`. For a match to succeed there must be GRs associated with the sentence that match each part of the pattern. Each argument matches either anything at all (*), the "current" adjective (˜), an empty GR argument (_), a `[word;id;part-of-speech]` 3-tuple or a numeric id. In a successful match, equal ids in different parts of the pattern must match the same word position, and distinct ids must match different positions.

The various patterns are arranged in a tree, where a parent node contains the elements common to all of its children. This kind of once-only representation of particular features, together with the successive refinements provided by child nodes reflects the organization of inheritance-based lexica. The inheritance structure naturally involves multiple inheritance, since each frame typically includes multiple features (such as the presence of a `to-infinitive` complement or an expletive subject argument) inherited from abstract parent classes, and each feature is instantiated in several frames.

The tree structure also improves the efficiency of the pattern matching process, which then occurs in stages: at each matching node the classifier attempts to match a set of relations with each child pattern to yield a substitution that subsumes the substitution resulting from the parent match.

Both the patterns and the pattern language itself underwent successive refinements after investigation of the performance on training data made it increasingly clear what sort of distinctions were useful to express. The initial pattern language had no slots; it was easy to understand and implement, but insufficiently expressive. The final refinement was the addition of ordering constraints between instantiated slots, which are indispensable for detecting, e.g., extraposition.

unspecified	285	improbable	350
unsure	570	doubtful	1147
generous	2052	sure	13591
difficult	18470	clear	19617
important	33303		

Table 1: Test adjectives and frequencies in the BNC

4 Experimental Evaluation

4.1 Data

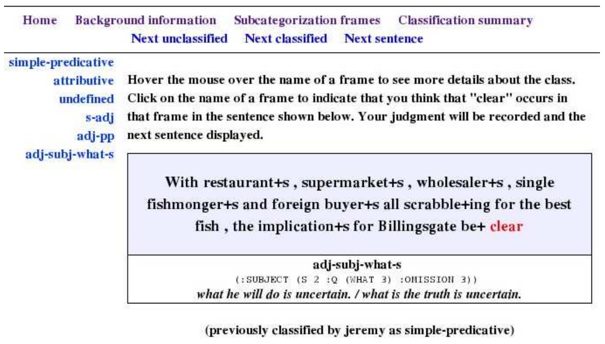
In order to evaluate the system we selected a set of 9 adjectives which between them could instantiate all of the frames. The test set was intentionally kept fairly small for these first experiments with adjectival SCF acquisition so that we could carry out a thorough evaluation of all the test instances. We excluded the adjectives used during development and adjectives with fewer than 200 instances in the corpus. The final test set, together with their frequencies in the tagged version of the BNC, is shown in table 1. For each adjective we extracted 200 sentences (evenly spaced throughout the BNC) which we processed using the SCF acquisition system described in the previous section.

4.2 Method

4.2.1 Annotation Tool and Gold Standard

Our gold standard was human-annotated data. Two annotators associated a SCF with each sentence/adjective pair in the test data. To alleviate the process we developed a program which first uses reliable heuristics to reduce the number of SCF choices and then allows the annotator to select the preferred choice with a single mouse click in a browser window. The heuristics reduced the average number of SCFs presented alongside each sentence from 30 to 9. Through the same browser interface we provided annotators with information and instructions (with links to COMLEX documentation), the ability to inspect and review previous decisions and decision summaries[5] and an option to record that partic-

[5]The varying number of SCFs presented to the user and the ability to revisit previous decisions precluded accurate measure-



Hover the mouse over the name of a frame to see more details about the class. Click on the name of a frame to indicate that you think that "clear" occurs in that frame in the sentence shown below. Your judgment will be recorded and the next sentence displayed.

With restaurant+s , supermarket+s , wholesaler+s , single fishmonger+s and foreign buyer+s all scrabble+ing for the best fish , the implication+s for Billingsgate be+ clear

adj-subj-what-s

```
(:SUBJECT (S 2 :Q (WHAT 3) :OMISSION 3))
```

what he will do is uncertain. / what is the truth is uncertain.

(previously classified by jeremy as simple-predicative)

Figure 6: Sample classification screen for web annotation tool

ular sentences could not be classified (which is useful for further system development, as discussed in section 5). A screenshot is shown in figure 6. The resulting annotation revealed 19 of the 30 SCFs in the test data.

4.2.2 Evaluation Measures

We use the standard evaluation metrics: type and token precision, recall and F-measure. *Token recall* is the proportion of annotated *(sentence, frame)* pairs that the system recovered correctly. *Token precision* is the proportion of classified *(sentence, frame)* pairs that were correct. *Type precision* and *type recall* are analogously defined for *(adjective, frame)* pairs. The F-measure ($\beta = 1$) is a weighted combination of precision and recall.

4.3 Results

Running the system on the test data yielded the results summarised in table 2. The greater expressiveness of the final pattern language resulted in a classifier that performed better than the "regression" versions which ignored either ordering constraints, or both ordering constraints and slots. As expected, removing features from the classifier translated directly into degraded accuracy. The performance of the best classifier (67.8% F-measure) is quite similar to that of the best current verbal SCF acquisition systems (e.g. (Korhonen, 2002)).

Results for individual adjectives are given in table 3. The first column shows the number of SCFs acquired for each adjective, ranging from 2 for *unspecified* to 11 for *doubtful*. Looking at the F-measure, the best performing adjectives are *unspecified*, *difficult* and *sure* (80%) and the worst performing *unsure* (50%) and and *improbable* (60%).

ments of inter-annotator agreement, but this was judged less important than the enhanced ease of use arising from the reduced set of choices.

Type performance

System	Precision	Recall	F
Final	69.6	66.1	67.8
No order constraints	67.3	62.7	64.9
No slots	62.7	51.4	56.5

Token performance

System	Precision	Recall	F
Final	63.0	70.5	66.5
No order constraints	58.8	68.3	63.2
No slots	58.3	67.6	62.6

Table 2: Overall performance of the classifier and of regression systems with restricted pattern-matching

There appears to be no obvious connection between performance figures and the number of acquired SCF types; differences are rather due to the difficulty of detecting individual SCF types — an issue directly related to data sparsity.

Despite the size of the BNC, 5 SCFs were not seen at all, either for the test adjectives or for any others. Frames involving *to-infinitive* complements were particularly rare: 4 such SCFs had no examples in the corpus and a further 3 occurred 5 times or fewer in the test data. It is more difficult to develop patterns for SCFs that occur infrequently, and the few instances of such SCFs are unlikely to include a set of GRs that is adequate for classification. The effect on the results was clear: of the 9 SCFs which the classifier did not correctly recognise at all, 4 occurred 5 times or fewer in the test data and a further 2 occurred 5–10 times.

The most common error made by the classifier was to mistake a complex frame (e.g. `adj-obj-for-to-inf`, or `to-inf-wh-adj`) for `simple-predicative`, which subsumes all such frames. This occurred whenever the GRs emitted by the parser failed to include any information about the complements of the adjective.

5 Discussion

Data sparsity is perhaps the greatest hindrance both to recovering adjectival subcategorization and to lexical acquisition in general. In the future, we plan to carry out experiments with a larger set of adjec-

Adjective	SCFs	Precision	Recall	F-measure
unspecified	2	66.7	100.0	80.0
generous	3	60.0	100.0	75.0
improbable	5	60.0	60.0	60.0
unsure	6	50.0	50.0	50.0
important	7	55.6	71.4	62.5
clear	8	83.3	62.5	71.4
difficult	8	85.7	75.0	80.0
sure	9	100.0	66.7	80.0
doubtful	11	66.7	54.5	60.0

Table 3: SCF count and classifier performance for each adjective.

tives using more data (possibly from several corpora and the web) to determine how severe this problem is for adjectives. One possible way to address the problem is to smooth the acquired SCF distributions using SCF "back-off" (probability) estimates based on lexical classes of adjectives in the manner proposed by (Korhonen, 2002). This helps to correct the acquired distributions and to detect low frequency and unseen SCFs.

However, our experiment also revealed other problems which require attention in the future. One such is that GRs output by RASP (the version we used in our experiments) do not retain certain distinctions which are essential for distinguishing particular SCFs. For example, a sentential complement of an adjective with a *that*-complementiser should be annotated with `ccomp(that, adjective, verbal-head)`, but this relation (with `that` as the *type* argument) does not occur in the parsed BNC. As a consequence the classifier is unable to distinguish the frame.

Another problem arises from the fact that our current classifier operates on a predefined set of SCFs. The COMLEX SCFs, from which ours were derived, are extremely incomplete. Almost a quarter (477 of 1931) of sentences were annotated as "undefined". For example, while there are SCFs for sentential and infinitival complement in subject position with *what*[6], there is no SCF for the case with a `what`-prefixed complement in object position, where the subject is an NP. The lack is especially perplexing, because COMLEX does include the corresponding SCFs for verbs. There is a frame for "He wondered what to do" (`what-to-inf`), but none for "He was unsure what to do".

[6](`adj-subj-what-s`: "What he will do is uncertain"; `adj-subj-what-to-inf`: "What to do was unclear"), together with the extraposed versions (`extrap-adj-what-s` and `extrap-adj-what-to-inf`).

While we can easily extend the current frameset by looking for further SCF types from dictionaries and from among the corpus occurrences labelled by our annotators as unclassified, we also plan to extend the classifier to automatically induce previously unseen frames from data. A possible approach is to use restricted generalization on sets of GRs to group similar sentences together. Generalization (*anti-unification*) is an intersection operation on two structures which retains the features common to both; generalization over the sets of GRs associated with the sentences which instantiate a particular frame can produce a pattern such as we used for classification in the experiments described above. This approach also offers the possibility of associating confidence levels with each pattern, corresponding to the degree to which the generalized pattern captures the features common to the members of the associated class. It is possible that frames could be induced by grouping sentences according to the "best" (e.g. most information-preserving) generalizations for various combinations, but it is not clear how this can be implemented with acceptable efficiency.

The hierarchical approach described in this paper may also helpful in the discovery of new frames: missing combinations of parent classes can be explored readily, and it may be possible to combine the various features in an SCF feature structure to generate example sentences which a human could then inspect to judge grammaticality.

6 Conclusion

We have described a novel system for automatically acquiring adjectival subcategorization and associated frequency information from corpora, along with an annotation tool for producing training and test data for the task. The acquisition system, which is capable of distinguishing 30 SCF types, performs sophisticated pattern matching on sets of GRs produced by a robust statistical parser. The information provided by GRs closely matches the structure that subcategorization acquisition seeks to recover. The figures reported demonstrate the feasibility of the approach: our classifier achieved 70% type precision and 66% type recall on the test data. The discussion suggests several ways in which the system may be improved, refined and extended in the future.

Acknowledgements

We would like to thank Ann Copestake for all her help during this work.

References

B. Boguraev, J. Carroll, E. Briscoe, D. Carter, and C. Grover. 1987. The derivation of a grammatically-indexed lexicon from the Longman Dictionary of Contemporary English. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 193–200, Stanford, CA.

Michael R. Brent. 1991. Automatic acquisition of subcategorization frames from untagged text. In *Meeting of the Association for Computational Linguistics*, pages 209–214.

E. J. Briscoe and J. Carroll. 1997. Automatic Extraction of Subcategorization from Corpora. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington DC, USA.

E. Briscoe and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1499–1504, Las Palmas, Canary Islands, May.

E. Briscoe, J. Carroll, Jonathan Graham, and Ann Copestake. 2002. Relational evaluation schemes. In *Proceedings of the Beyond PARSEVAL Workshop at the 3rd International Conference on Language Resources and Evaluation*, pages 4–8, Las Palmas, Gran Canaria.

Lou Burnard, 1995. *The BNC Users Reference Guide*. British National Corpus Consortium, Oxford, May.

J. Carroll and E. Briscoe. 2002. High precision extraction of grammatical relations. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 134–140, Taipei, Taiwan.

Glenn Carroll and Mats Rooth. 1998. Valence induction with a head-lexicalized pcfg. In *Proc. of the 3rd Conference on Empirical Methods in Natural Language Processing*, Granada, Spain.

J. Carroll, E. Briscoe, and A. Sanfi lippo. 1998a. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain.

John Carroll, Guido Minnen, and Edward Briscoe. 1998b. Can Subcategorisation Probabilities Help a Statistical Parser? In *Proceedings of the 6th ACL/SIGDAT Workshop on Very Large Corpora*, pages 118–126, Montreal, Canada. Association for Computational Linguistics.

Eva Esteve Ferrer. 2004. Towards a Semantic Classifi cation of Spanish Verbs Based on Subcategorisation Information. In *ACL Student Research Workshop*, Barcelona, Spain.

Dan Flickinger and John Nerbonne. 1992. Inheritance and complementation: A case study of easy adjectives and related nouns. *Computational Linguistics*, 18(3):269–309.

Daisuke Kawahara and Sadao Kurohashi. 2002. Fertilization of Case Frame Dictionary for Robust Japanese Case Analysis. In *19th International Conference on Computational Linguistics*.

Anna Korhonen, Yuval Krymolowski, and Zvika Marx. 2003. Clustering Polysemic Subcategorization Frame Distributions Semantically. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 64–71, Sapporo, Japan.

Anna Korhonen. 2002. *Subcategorization acquisition*. Ph.D. thesis, University of Cambridge Computer Laboratory, February.

Catherine Macleod, Ralph Grishman, and Adam Meyers, 1998. COMLEX *Syntax Reference Manual*. Computer Science Department, New York University.

Christopher D. Manning. 1993. Automatic Acquisition of a Large Subcategorization Dictionary from Corpora. In *Meeting of the Association for Computational Linguistics*, pages 235–242.

S. Schulte im Walde and C. Brew. 2002. Inducing german semantic verb classes from purely syntactic subcategorisation information. In *40th Annual Meeting of the Association for Computational Linguistics*, Philadephia, USA.

Mihai Surdeanu, Sanda Harabagiu, JohnWilliams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo.

Susanne Rohen Wolff, Catherine Macleod, and Adam Meyers, 1998. COMLEX *Word Classes Manual*. Computer Science Department, New York University , June.

Randomized Algorithms and NLP: Using Locality Sensitive Hash Function for High Speed Noun Clustering

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292.
{ravichan, pantel, hovy}@ISI.EDU

Abstract

In this paper, we explore the power of randomized algorithm to address the challenge of working with very large amounts of data. We apply these algorithms to generate noun similarity lists from 70 million pages. We reduce the running time from quadratic to practically linear in the number of elements to be computed.

1 Introduction

In the last decade, the field of Natural Language Processing (NLP), has seen a surge in the use of corpus motivated techniques. Several NLP systems are modeled based on empirical data and have had varying degrees of success. Of late, however, corpus-based techniques seem to have reached a plateau in performance. Three possible areas for future research investigation to overcoming this plateau include:
1. Working with large amounts of data (Banko and Brill, 2001)
2. Improving semi-supervised and unsupervised algorithms.
3. Using more sophisticated feature functions.

The above listing may not be exhaustive, but it is probably not a bad bet to work in one of the above directions. In this paper, we investigate the first two avenues. Handling terabytes of data requires more efficient algorithms than are currently used in NLP. We propose a web scalable solution to clustering nouns, which employs randomized algorithms. In doing so, we are going to explore the literature and techniques of randomized algorithms. All clustering algorithms make use of some distance similarity (e.g., cosine similarity) to measure pair wise distance between sets of vectors. Assume that we are given n points to cluster with a maximum of k features. Calculating the full similarity matrix would take time complexity n^2k. With large amounts of data, say n in the order of millions or even billions, having an n^2k algorithm would be very infeasible. To be scalable, we ideally want our algorithm to be proportional to nk.

Fortunately, we can borrow some ideas from the Math and Theoretical Computer Science community to tackle this problem. The crux of our solution lies in defining Locality Sensitive Hash (LSH) functions. LSH functions involve the creation of short signatures (fingerprints) for each vector in space such that those vectors that are closer to each other are more likely to have similar fingerprints. LSH functions are generally based on randomized algorithms and are probabilistic. We present LSH algorithms that can help reduce the time complexity of calculating our distance similarity atrix to nk.

Rabin (1981) proposed the use of hash functions from random irreducible polynomials to create short fingerprint representations for very large strings. These hash function had the nice property that the fingerprint of two identical strings had the same fingerprints, while dissimilar strings had different fingerprints with a very small probability of collision. Broder (1997) first introduced LSH. He proposed the use of Min-wise independent functions to create fingerprints that preserved the Jaccard similarity between every pair of vectors. These techniques are used today, for example, to eliminate duplicate web pages. Charikar (2002) proposed the use of random hyperplanes to generate an LSH function that preserves the cosine similarity between every pair of vectors. Interestingly, cosine similarity is widely used in NLP for various applications such as clustering.

In this paper, we perform high speed similarity list creation for nouns collected from a huge web corpus. We linearize this step by using the LSH proposed by Charikar (2002). This reduction in complexity of similarity computation makes it possible to address vastly larger datasets, at the cost, as shown in Section 5, of only little reduction in accuracy. In our experiments, we generate a similarity list for each noun extracted from 70 million page web corpus. Although the NLP community has begun experimenting with the web, we know of no work in published literature that has applied complex language analysis beyond IR and simple surface-level pattern matching.

2 Theory

The core theory behind the implementation of fast cosine similarity calculation can be divided into two parts: **1.** Developing LSH functions to create signatures; **2.** Using fast search algorithm to find nearest neighbors. We describe these two components in greater detail in the next subsections.

2.1 LSH Function Preserving Cosine Similarity

We first begin with the formal definition of cosine similarity.

Definition: Let u and v be two vectors in a k dimensional hyperplane. Cosine similarity is defined as the cosine of the angle between them: $cos(\theta(u, v))$. We can calculate $cos(\theta(u, v))$ by the following formula:

$$cos(\theta(u, v)) = \frac{|u.v|}{|u||v|} \quad (1)$$

Here $\theta(u, v)$ is the angle between the vectors u and v measured in radians. $|u.v|$ is the scalar (dot) product of u and v, and $|u|$ and $|v|$ represent the length of vectors u and v respectively.

The LSH function for cosine similarity as proposed by Charikar (2002) is given by the following theorem:

Theorem: Suppose we are given a collection of vectors in a k dimensional vector space (as written as R^k). Choose a family of hash functions as follows: Generate a spherically symmetric random vector r of unit length from this k dimensional space. We define a hash function, h_r, as:

$$h_r(u) = \begin{cases} 1 & : \ r.u \geq 0 \\ 0 & : \ r.u < 0 \end{cases} \quad (2)$$

Then for vectors u and v,

$$Pr[h_r(u) = h_r(v)] = 1 - \frac{\theta(u, v)}{\pi} \quad (3)$$

Proof of the above theorem is given by Goemans and Williamson (1995). We rewrite the proof here for clarity. The above theorem states that the probability that a random hyperplane separates two vectors is directly proportional to the angle between the two vectors (i,e., $\theta(u, v)$). By symmetry, we have $Pr[h_r(u) \neq h_r(v)] = 2Pr[u.r \geq 0, v.r < 0]$. This corresponds to the intersection of two half spaces, the dihedral angle between which is θ. Thus, we have $Pr[u.r \geq 0, v.r < 0] = \theta(u, v)/2\pi$. Proceeding we have $Pr[h_r(u) \neq h_r(v)] = \theta(u, v)/\pi$ and $Pr[h_r(u) = h_r(v)] = 1 - \theta(u, v)/\pi$. This completes the proof.

Hence from equation 3 we have,

$$cos(\theta(u, v)) = cos((1 - Pr[h_r(u) = h_r(v)])\pi) \quad (4)$$

This equation gives us an alternate method for finding cosine similarity. Note that the above equation is probabilistic in nature. Hence, we generate a large (d) number of random vectors to achieve the process. Having calculated $h_r(u)$ with d random vectors for each of the vectors u, we apply equation 4 to find the cosine distance between two vectors. As we generate more number of random vectors, we can estimate the cosine similarity between two vectors more accurately. However, in practice, the number (d) of random vectors required is highly domain dependent, i.e., it depends on the value of the total number of vectors (n), features (k) and the way the vectors are distributed. Using d random vectors, we can represent each vector by a bit stream of length d.

Carefully looking at equation 4, we can observe that $Pr[h_r(u) = h_r(v)] = 1 - (hamming\ distance)/d$[1] . Thus, the above theorem, converts the problem of finding cosine distance between two vectors to the problem of finding hamming distance between their bit streams (as given by equation 4). Finding hamming distance between two bit streams is faster and highly memory efficient. Also worth noting is that this step could be considered as dimensionality reduction wherein we reduce a vector in k dimensions to that of d bits while still preserving the cosine distance between them.

2.2 Fast Search Algorithm

To calculate the fast hamming distance, we use the search algorithm PLEB (Point Location in Equal Balls) first proposed by Indyk and Motwani (1998). This algorithm was further improved by Charikar (2002). This algorithm involves random permutations of the bit streams and their sorting to find the vector with the closest hamming distance. The algorithm given in Charikar (2002) is described to find the nearest neighbor for a given vector. We modify it so that we are able to find the top B closest neighbor for each vector. We omit the math of this algorithm but we sketch its procedural details in the next section. Interested readers are further encouraged to read Theorem 2 from Charikar (2002) and Section 3 from Indyk and Motwani (1998).

3 Algorithmic Implementation

In the previous section, we introduced the theory for calculation of fast cosine similarity. We implement it as follows:

1. Initially we are given n vectors in a huge k dimensional space. Our goal is to find all pairs of vectors whose cosine similarity is greater than a particular threshold.

2. Choose d number of ($d << k$) unit random vectors $\{r_0, r_1, ..., r_d\}$ each of k dimensions.

   A k dimensional unit random vector, in general, is generated by independently sampling a Gaussian function with mean 0 and variance 1, k number of times. Each of the k samples is used to assign one dimension to the random vector. We generate a random number from a Gaussian distribution by using Box-Muller transformation (Box and Muller, 1958).

3. For every vector u, we determine its signature by using the function $h_r(u)$ (as given by equation 4). We can represent the signature of vector u as: $\bar{u} = \{h_{r1}(u), h_{r2}(u), ..., h_{rd}(u)\}$. Each vector is thus represented by a set of a bit streams of length d. Steps 2 and 3 takes $O(nk)$ time (We can assume d to be a constant since $d << k$).

4. The previous step gives n vectors, each of them represented by d bits. For calculation of fast hamming distance, we take the original bit index of all vectors and randomly permute them (see Appendix A for more details on random permutation functions). A random permutation can be considered as random jumbling of the bits of each vector[2]. A random permutation function can be approximated by the following function:

$$\pi(x) = (ax + b) \text{mod p} \tag{5}$$

   where, p is prime and $0 < a < p$, $0 \leq b < p$, and a and b are chosen at random.

   We apply q different random permutation for every vector (by choosing random values for a and b, q number of times). Thus for every vector we have q different bit permutations for the original bit stream.

5. For each permutation function π, we lexicographically sort the list of n vectors (whose bit streams are permuted by the function π) to obtain a sorted list. This step takes $O(nlogn)$ time. (We can assume q to be a constant).

6. For each sorted list (performed after applying the random permutation function π), we calculate the hamming distance of every vector with

[1]Hamming distance is the number of bits which differ between two binary strings.

[2]The jumbling is performed by a mapping of the bit index as directed by the random permutation function. For a given permutation, we reorder the bit indexes of all vectors in similar fashion. This process could be considered as column reording of bit vectors.

B of its closest neighbors in the sorted list. If the hamming distance is below a certain predetermined threshold, we output the pair of vectors with their cosine similarity (as calculated by equation 4). Thus, B is the beam parameter of the search. This step takes $O(n)$, since we can assume B, q, d to be a constant.

Why does the fast hamming distance algorithm work? The intuition is that the number of bit streams, d, for each vector is generally smaller than the number of vectors n (ie. $d << n$). Thus, sorting the vectors lexicographically after jumbling the bits will likely bring vectors with lower hamming distance closer to each other in the sorted lists.

Overall, the algorithm takes $O(nk + nlogn)$ time. However, for noun clustering, we generally have the number of nouns, n, smaller than the number of features, k. (i.e., $n < k$). This implies $logn << k$ and $nlogn << nk$. Hence the time complexity of our algorithm is $O(nk + nlogn) \approx O(nk)$. This is a huge saving from the original $O(n^2k)$ algorithm. In the next section, we proceed to apply this technique for generating noun similarity lists.

4 Building Noun Similarity Lists

A lot of work has been done in the NLP community on clustering words according to their meaning in text (Hindle, 1990; Lin, 1998). The basic intuition is that words that are similar to each other tend to occur in similar contexts, thus linking the semantics of words with their lexical usage in text. One may ask why is clustering of words necessary in the first place? There may be several reasons for clustering, but generally it boils down to one basic reason: if the words that occur rarely in a corpus are found to be distributionally similar to more frequently occurring words, then one may be able to make better inferences on rare words.

However, to unleash the real power of clustering one has to work with large amounts of text. The NLP community has started working on noun clustering on a few gigabytes of newspaper text. But with the rapidly growing amount of raw text available on the web, one could improve clustering performance by carefully harnessing its power. A core component of most clustering algorithms used in the NLP community is the creation of a similarity matrix. These algorithms are of complexity $O(n^2k)$, where n is the number of unique nouns and k is the feature set length. These algorithms are thus not readily scalable, and limit the size of corpus manageable in practice to a few gigabytes. Clustering algorithms for words generally use the cosine distance for their similarity calculation (Salton and McGill, 1983). Hence instead of using the usual naive cosine distance calculation between every pair of words we can use the algorithm described in Section 3 to make noun clustering web scalable.

To test our algorithm we conduct similarity based experiments on 2 different types of corpus: **1.** Web Corpus (70 million web pages, 138GB), **2.** Newspaper Corpus (6 GB newspaper corpus)

4.1 Web Corpus

We set up a spider to download roughly 70 million web pages from the Internet. Initially, we use the links from Open Directory project[3] as seed links for our spider. Each webpage is stripped of HTML tags, tokenized, and sentence segmented. Each document is language identified by the software TextCat[4] which implements the paper by Cavnar and Trenkle (1994). We retain only English documents. The web contains a lot of duplicate or near-duplicate documents. Eliminating them is critical for obtaining better representation statistics from our collection. The problem of identifying near duplicate documents in linear time is not trivial. We eliminate duplicate and near duplicate documents by using the algorithm described by Kolcz et al. (2004). This process of duplicate elimination is carried out in linear time and involves the creation of signatures for each document. Signatures are designed so that duplicate and near duplicate documents have the same signature. This algorithm is remarkably fast and has high accuracy. This entire process of removing non English documents and duplicate (and near-duplicate) documents reduces our document set from 70 million web pages to roughly 31 million web pages. This represents roughly 138GB of uncompressed text.

We identify all the nouns in the corpus by using a noun phrase identifier. For each noun phrase, we identify the context words surrounding it. Our context window length is restricted to two words to

[3]http://www.dmoz.org/

[4]http://odur.let.rug.nl/∼vannoord/TextCat/

Table 1: Corpus description

Corpus	Newspaper	Web
Corpus Size	6GB	138GB
Unique Nouns	65,547	655,495
Feature size	940,154	1,306,482

the left and right of each noun. We use the context words as features of the noun vector.

4.2 Newspaper Corpus

We parse a 6 GB newspaper (TREC9 and TREC2002 collection) corpus using the dependency parser Minipar (Lin, 1994). We identify all nouns. For each noun we take the grammatical context of the noun as identified by Minipar[5]. We do not use grammatical features in the web corpus since parsing is generally not easily web scalable. This kind of feature set does not seem to affect our results. Curran and Moens (2002) also report comparable results for Minipar features and simple word based proximity features. Table 1 gives the characteristics of both corpora. Since we use grammatical context, the feature set is considerably larger than the simple word based proximity feature set for the newspaper corpus.

4.3 Calculating Feature Vectors

Having collected all nouns and their features, we now proceed to construct feature vectors (and values) for nouns from both corpora using mutual information (Church and Hanks, 1989). We first construct a frequency count vector $C(e) = (c_{e1}, c_{e2}, ..., c_{ek})$, where k is the total number of features and c_{ef} is the frequency count of feature f occurring in word e. Here, c_{ef} is the number of times word e occurred in context f. We then construct a mutual information vector $MI(e) = (mi_{e1}, mi_{e2}, ..., mi_{ek})$ for each word e, where mi_{ef} is the pointwise mutual information between word e and feature f, which is defined as:

$$mi_{ef} = log \frac{\frac{c_{ef}}{N}}{\sum_{i=1}^{n} \frac{c_{if}}{N} \times \sum_{j=1}^{k} \frac{c_{ej}}{N}} \tag{6}$$

where n is the number of words and $N = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}$ is the total frequency count of all features of all words.

Having thus obtained the feature representation of each noun we can apply the algorithm described in Section 3 to discover similarity lists. We report results in the next section for both corpora.

5 Evaluation

Evaluating clustering systems is generally considered to be quite difficult. However, we are mainly concerned with evaluating the quality and speed of our high speed randomized algorithm. The web corpus is used to show that our framework is web-scalable, while the newspaper corpus is used to compare the output of our system with the similarity lists output by an existing system, which are calculated using the traditional formula as given in equation 1. For this base comparison system we use the one built by Pantel and Lin (2002). We perform 3 kinds of evaluation: **1.** Performance of Locality Sensitive Hash Function; **2.** Performance of fast Hamming distance search algorithm; **3.** Quality of final similarity lists.

5.1 Evaluation of Locality sensitive Hash function

To perform this evaluation, we randomly choose 100 nouns (vectors) from the web collection. For each noun, we calculate the cosine distance using the traditional slow method (as given by equation 1), with all other nouns in the collection. This process creates similarity lists for each of the 100 vectors. These similarity lists are cut off at a threshold of 0.15. These lists are considered to be the gold standard test set for our evaluation.

For the above 100 chosen vectors, we also calculate the cosine similarity using the randomized approach as given by equation 4 and calculate the mean squared error with the gold standard test set using the following formula:

$$error_{av} = \sqrt{\sum_i (CS_{real,i} - CS_{calc,i})^2 / total} \tag{7}$$

where $CS_{real,i}$ and $CS_{calc,i}$ are the cosine similarity scores calculated using the traditional (equation 1) and randomized (equation 4) technique re-

[5] We perform this operation so that we can compare the performance of our system to that of Pantel and Lin (2002).

Table 2: Error in cosine similarity

Number of random vectors d	Average error in cosine similarity	Time (in hours)
1	1.0000	0.4
10	0.4432	0.5
100	0.1516	3
1000	0.0493	24
3000	0.0273	72
10000	0.0156	241

spectively. i is the index over all pairs of elements that have $CS_{real,i} >= 0.15$

We calculate the error ($error_{av}$) for various values of d, the total number of unit random vectors r used in the process. The results are reported in Table 2[6]. As we generate more random vectors, the error rate decreases. For example, generating 10 random vectors gives us a cosine error of 0.4432 (which is a large number since cosine similarity ranges from 0 to 1.) However, generation of more random vectors leads to reduction in error rate as seen by the values for 1000 (0.0493) and 10000 (0.0156). But as we generate more random vectors the time taken by the algorithm also increases. We choose $d = 3000$ random vectors as our optimal (time-accuracy) cut off. It is also very interesting to note that by using only 3000 bits for each of the 655,495 nouns, we are able to measure cosine similarity between every pair of them to within an average error margin of 0.027. This algorithm is also highly memory efficient since we can represent every vector by only a few thousand bits. Also the randomization process makes the the algorithm easily parallelizable since each processor can independently contribute a few bits for every vector.

5.2 Evaluation of Fast Hamming Distance Search Algorithm

We initially obtain a list of bit streams for all the vectors (nouns) from our web corpus using the randomized algorithm described in Section 3 (Steps 1 to 3). The next step involves the calculation of hamming distance. To evaluate the quality of this search algorithm we again randomly choose 100 vectors (nouns) from our collection. For each of these 100 vectors we manually calculate the hamming distance with all other vectors in the collection. We only retain those pairs of vectors whose cosine distance (as manually calculated) is above 0.15. This similarity list is used as the gold standard test set for evaluating our fast hamming search.

We then apply the fast hamming distance search algorithm as described in Section 3. In particular, it involves steps 3 to 6 of the algorithm. We evaluate the hamming distance with respect to two criteria: **1.** Number of bit index random permutations functions q; **2.** Beam search parameter B.

For each vector in the test collection, we take the top N elements from the gold standard similarity list and calculate how many of these elements are actually discovered by the fast hamming distance algorithm. We report the results in Table 3 and Table 4 with beam parameters of ($B = 25$) and ($B = 100$) respectively. For each beam, we experiment with various values for q, the number of random permutation function used. In general, by increasing the value for beam B and number of random permutation q , the accuracy of the search algorithm increases. For example in Table 4 by using a beam $B = 100$ and using 1000 random bit permutations, we are able to discover 72.8% of the elements of the Top 100 list. However, increasing the values of q and B also increases search time. With a beam (B) of 100 and the number of random permutations equal to 100 (i.e., $q = 1000$) it takes 570 hours of processing time on a single Pentium IV machine, whereas with $B = 25$ and $q = 1000$, reduces processing time by more than 50% to 240 hours.

We could not calculate the total time taken to build noun similarity list using the traditional technique on the entire corpus. However, we estimate that its time taken would be at least 50,000 hours (and perhaps even more) with a few of Terabytes of disk space needed. This is a very rough estimate. The experiment was infeasible. This estimate assumes the widely used reverse indexing technique, where in one compares only those vector pairs that have at least one feature in common.

5.3 Quality of Final Similarity Lists

For evaluating the quality of our final similarity lists, we use the system developed by Pantel and Lin (2002) as gold standard on a much smaller data set. We use the same 6GB corpus that was used for train-

[6]The time is calculated for running the algorithm on a single Pentium IV processor with 4GB of memory

Table 3: Hamming search accuracy (Beam $B = 25$)

Random permutations q	Top 1	Top 5	Top 10	Top 25	Top 50	Top 100
25	6.1%	4.9%	4.2%	3.1%	2.4%	1.9%
50	6.1%	5.1%	4.3%	3.2%	2.5%	1.9%
100	11.3%	9.7%	8.2%	6.2%	5.7%	5.1%
500	44.3%	33.5%	30.4%	25.8%	23.0%	20.4%
1000	58.7%	50.6%	48.8%	45.0%	41.0%	37.2%

Table 4: Hamming search accuracy (Beam $B = 100$)

Random permutations q	Top 1	Top 5	Top 10	Top 25	Top 50	Top 100
25	9.2%	9.5%	7.9%	6.4%	5.8%	4.7%
50	15.4%	17.7%	14.6%	12.0%	10.9%	9.0%
100	27.8%	27.2%	23.5%	19.4%	17.9%	16.3%
500	73.1%	67.0%	60.7%	55.2%	53.0%	50.5%
1000	87.6%	84.4%	82.1%	78.9%	75.8%	72.8%

ing by Pantel and Lin (2002) so that the results are comparable. We randomly choose 100 nouns and calculate the top N elements closest to each noun in the similarity lists using the randomized algorithm described in Section 3. We then compare this output to the one provided by the system of Pantel and Lin (2002). For every noun in the top N list generated by our system we calculate the percentage overlap with the gold standard list. Results are reported in Table 5. The results shows that we are able to retrieve roughly 70% of the gold standard similarity list. In Table 6, we list the top 10 most similar words for some nouns, as examples, from the web corpus.

6 Conclusion

NLP researchers have just begun leveraging the vast amount of knowledge available on the web. By searching IR engines for simple surface patterns, many applications ranging from word sense disambiguation, question answering, and mining semantic resources have already benefited. However, most language analysis tools are too infeasible to run on the scale of the web. A case in point is generating noun similarity lists using co-occurrence statistics, which has quadratic running time on the input size. In this paper, we solve this problem by presenting a randomized algorithm that linearizes this task and limits memory requirements. Experiments show that our method generates cosine similarities between pairs of nouns within a score of 0.03.

In many applications, researchers have shown that more data equals better performance (Banko and Brill, 2001; Curran and Moens, 2002). Moreover, at the web-scale, we are no longer limited to a snapshot in time, which allows broader knowledge to be learned and processed. Randomized algorithms provide the necessary speed and memory requirements to tap into terascale text sources. We hope that randomized algorithms will make other NLP tools feasible at the terascale and we believe that many algorithms will benefit from the vast coverage of our newly created noun similarity list.

Acknowledgement

We wish to thank USC Center for High Performance Computing and Communications (HPCC) for helping us use their cluster computers.

References

Banko, M. and Brill, E. 2001. Mitigating the paucity of dataproblem. In Proceedings of *HLT*. 2001. San Diego, CA.

Box, G. E. P. and M. E. Muller 1958. *Ann. Math. Stat.* 29, 610–611.

Broder, Andrei 1997. On the Resemblance and Containment of Documents. Proceedings of the *Compression and Complexity of Sequences*.

Cavnar, W. B. and J. M. Trenkle 1994. N-Gram-Based Text Categorization. In Proceedings of Third Annual Symposium on *Document Analysis and Information Retrieval*, Las Vegas, NV, UNLV Publications/Reprographics, 161–175.

Table 5: Final Quality of Similarity Lists

	Top 1	Top 5	Top 10	Top 25	Top 50	Top 100
Accuracy	70.7%	71.9%	72.2%	71.7%	71.2%	71.1%

Table 6: Sample Top 10 Similarity Lists

JUST DO IT	computer science	TSUNAMI	Louis Vuitton	PILATES
HAVE A NICE DAY	mechanical engineering	tidal wave	PRADA	Tai Chi
FAIR AND BALANCED	electrical engineering	LANDSLIDE	Fendi	Cardio
POWER TO THE PEOPLE	chemical engineering	EARTHQUAKE	Kate Spade	SHIATSU
NEVER AGAIN	Civil Engineering	volcanic eruption	VUITTON	Calisthenics
NO BLOOD FOR OIL	ECONOMICS	HAILSTORM	BURBERRY	Ayurveda
KINGDOM OF HEAVEN	ENGINEERING	Typhoon	GUCCI	Acupressure
If Texas Wasn't	Biology	Mudslide	Chanel	Qigong
BODY OF CHRIST	environmental science	windstorm	Dior	FELDENKRAIS
WE CAN	PHYSICS	HURRICANE	Ferragamo	THERAPEUTIC TOUCH
Weld with your mouse	information science	DISASTER	Ralph Lauren	Reflexology

Charikar, Moses 2002. Similarity Estimation Techniques from Rounding Algorithms In Proceedings of the *34th Annual ACM Symposium on Theory of Computing*.

Church, K. and Hanks, P. 1989. Word association norms, mutual information, and lexicography. In Proceedings of *ACL-89*. pp. 76–83. Vancouver, Canada.

Curran, J. and Moens, M. 2002. Scaling context space. In Proceedings of *ACL-02* pp 231–238, Philadelphia, PA.

Goemans, M. X. and D. P. Williamson 1995. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *JACM* 42(6): 1115–1145.

Hindle, D. 1990. Noun classification from predicate-argument structures. In Proceedings of ACL-90. pp. 268–275. Pittsburgh, PA.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In Proceedings of *COLING/ACL-98*. pp. 768–774. Montreal, Canada.

Indyk, P., Motwani, R. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality Proceedings of *30th STOC*, 604–613.

A. Kolcz, A. Chowdhury, J. Alspector 2004. Improved robustness of signature-based near-replica detection via lexicon randomization. Proceedings of *ACM-SIGKDD* (2004).

Lin, D. 1994 Principar - an efficient, broad-coverage, principle-based parser. Proceedings of *COLING-94*, pp. 42–48. Kyoto, Japan.

Pantel, Patrick and Dekang Lin 2002. Discovering Word Senses from Text. In Proceedings of *SIGKDD-02*, pp. 613–619. Edmonton, Canada

Rabin, M. O. 1981. Fingerprinting by random polynomials. Center for research in Computing technology , Harvard University, *Report* TR-15-81.

Salton, G. and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

Appendix A. Random Permutation Functions

We define $[n] = \{0, 1, 2, ..., n-1\}$.
$[n]$ can thus be considered as a set of integers from 0 to $n-1$.
Let $\pi : [n] \rightarrow [n]$ be a permutation function chosen at random from the set of all such permutation functions.
Consider $\pi : [4] \rightarrow [4]$.
A permutation function π is a one to one mapping from the set of $[4]$ to the set of $[4]$.
Thus, one possible mapping is:
$\pi : \{0, 1, 2, 3\} \rightarrow \{3, 2, 1, 0\}$
Here it means: $\pi(0) = 3$, $\pi(1) = 2$, $\pi(2) = 1$, $\pi(3) = 0$
Another possible mapping would be:
$\pi : \{0, 1, 2, 3\} \rightarrow \{3, 0, 1, 2\}$
Here it means: $\pi(0) = 3$, $\pi(1) = 0$, $\pi(2) = 1$, $\pi(3) = 2$

Thus for the set $[4]$ there would be $4! = 4*3*2 = 24$ possibilities. In general, for a set $[n]$ there would be $n!$ unique permutation functions. Choosing a random permutation function amounts to choosing one of $n!$ such functions at random.

Author Index