10th Conference of the Pacific Association for Computational Linguistics

pacling 2007

19–21 September, 2007
University of Melbourne
Melbourne, Australia

Sponsors:

©2007 Pacific Association for Computational Linguistics

This work is licensed under a Creative Commons Attribution 3.0 License
(http://creativecommons.org/licenses/by/3.0/).

Order copies of these proceedings from:

Timothy Baldwin
Dept of Computer Science and Software Engineering
University of Melbourne
Victoria 3010, Australia
Tel: +61 3 8344 1363
Fax: +61 3 9348 1184
tim@csse.unimelb.edu.au

Introduction

This is the tenth meeting of the Pacific Association for Computational Linguistics. It is a pleasure to return to Melbourne, the site of the first such meeting. We hope that this meeting will help to bring together language technology workers from the entire pacific region.

This year, we received 96 submissions, 55 from Asia (40 from Japan), 23 from Australia and New Zealand, 12 from Europe and 6 from North America. After a thorough reviewing process, we selected 39 papers to be presented at the conference. There were so many high quality papers that we organized the program into two parallel sessions.

The papers selected range over a variety of topics, including Machine Translation, Parsing, Named Entity Extraction and Terminology. In keeping with the theme of "Deep processing: Abstracting away from the surface", the two most common themes are Lexical Semantics (including WSD) and Question Answering.

We would like to thank all the authors for submitting their research and the members of the program committee for their careful reviews and useful suggestions to the authors. We would also like to thank the three invited speakers: Ann Copestake, Robert Dale, and Yuji Matsumoto.

In order to further the advancement of science by removing unnecessary legal and technical barriers to scientific collaboration and innovation all the papers have been released under the Creative Commons Attribution 3.0 unported license (`http://creativecommons.org/licenses/by/3.0/`). This allows others to copy, use, distribute, transmit and display the papers in these proceedings as well as to make and distribute derivative works, subject to proper attribution of authorship. It is our hope that these proceedings will thus become the raw material for further research.

We are grateful to the National Institute of Information and Communications Technology, Japan (NICT), National ICT Australia (NICTA), and the University of Melbourne for sponsoring PACLING-2007. In addition, the conference would not have been possible without the tireless efforts of the local chairs, Timothy Baldwin and David Martinez. Finally we would like to thank the PACLING President, Shun Ishizaki for his help in the overall organization.

Francis Bond
National Institute of Information and Communications Technology, Japan
PACLING 2007 Program Chair

Organizers

Program Chair:

Francis Bond (NICT, Japan)

Local Chairs:

Timothy Baldwin (University of Melbourne, Australia)
David Martinez (University of Melbourne, Australia)

Program Committee:

Emily Bender (University of Washington, USA)
Steven Bird (University of Melbourne, Australia)
Sandra Carberry (University of Delaware, USA)
Lawrence Cavedon (NICTA, Australia)
Nick Cercone (York University, Canada)
James Curran (University of Sydney, Australia)
Robert Dale (Macquarie University, Australia)
Rebecca Dridan (University of the Saarlands, Germany)
Kevin Duh (University of Washington, USA)
Tsutomu Endo (Kyushu Institute of Technology, Japan)
Laurel Fais (University of British Columbia, Canada)
Dan Flickinger (Stanford University, USA)
John Fry (San Jose State University, USA)
Sanae Fujita (NTT, Japan)
Chikara Hashimoto (Yamagata University, Japan)
Yoshihiko Hayashi (Osaka University, Japan)
Lars Hellan (NTNU, Norway)
Graeme Hirst (University of Toronto, Canada)
Young-Sook Hwang (ETRI, Korea)
Satoru Ikehara (Tottori University, Japan)
Kentaro Inui (Nara Institute of Science and Technology, Japan)
Pierre Isabelle (University of Quebec, Canada)
Shun Ishizaki (Keio University, Japan)
Daisuke Kawahara (NICT, Japan)
Jong-Bok Kim (Kyung Hee University, Korea)
Ali Knott (University of Otago, New Zealand)
Kiyoshi Kogure (ATR, Japan)
Rob Malouf (San Diego State University, USA)
Yuval Marom (GAPbuster Worldwide, Australia)
Hiroshi Masuichi (Fuji Xerox, Japan)
Robert Mercer (University of Western Ontario, Canada)
Adam Meyers (New York University, USA)

Diego Molla (Macquarie University, Australia)
Masaki Murata (NICT, Japan)
Akiyo Nadamoto (NICT, Japan)
Ryo Nagata (Hyogo University of Teacher Education, Japan)
Hiromi Nakaiwa (NTT Corp., Japan)
Tsuneko Nakazawa (Tokyo University, Japan)
Shigeko Nariyama (University of Melbourne, Australia)
Hwee-Tou Ng (National University of Singapore, Singapore)
Eric Nichols (NAIST, Japan)
Yoshihiko Nitta (Nihon University, Japan)
Stephan Oepen (University of Oslo, Norway)
Kentaro Ogura (NTT Software, Japan)
Cécile Paris (CSIRO, Australia)
David Powers (Flinders University, Australia)
Hiroshi Sakaki (Meisei University, Japan)
Ulrich Schaefer (DFKI, Germany)
Virach Sornlertlamvanich (NICT, Thailand)
Nicola Stokes (NICTA, Australia)
Thepchai Supnithi (NECTEC, Thailand)
Hisami Suzuki (Microsoft Research, USA)
Kumiko Tanaka-Ishii (University of Tokyo, Japan)
Takaaki Tanaka (NTT, Japan)
Enya-Kong Tan (Penang University, Malaysia)
Ben Waldron (NTNU, Norway)
Xiaohong Wu (NICT, Japan)
Kazuhide Yamamoto (Nagaoka University of Technology, Japan)
Jaehyung Yang (Kangnam University, Korea)
Yi Zhang (DFKI, Germany)
Yujie Zhang (NICT, Japan)

Invited Speakers:

Ann Copestake (University of Cambridge, UK)
Robert Dale (Macquarie University, Australia)
Yuji Matsumoto (NAIST, Japan)

Table of Contents

Applying Robust Semantics
Ann Copestake 1

Pinyomi: Dictionary lookup via orthographic associations
Lars Yencken, Zhihui Jin and Kumiko Tanaka-Ishii 13

Graphical query for linguistic treebanks
Steven Bird and Haejoong Lee 22

Corpus Statistics Approaches to Discriminating Among Near-Synonyms
Mary Gardiner and Mark Dras 31

Detecting Compositionality of English Verb-Particle Constructions using Semantic Similarity
Su Nam Kim and Timothy Baldwin 40

Sentence and Token Splitting Based On Conditional Random Fields
Katrin Tomanek, Joachim Wermter and Udo Hahn 49

Using Conditional Random Fields for Clause Splitting
Vinh Van Nguyen, Minh Le Nguyen and Akira Shimazu 58

Incorporating Global Information into Supervised Learning for Chinese Word Segmentation
Hai Zhao and Chunyu Kit 66

Combining Similarity-based Approaches for Semantic Tagging of Unknown Words
Fumiyo Fukumoto and Yoshimi Suzuki 75

Using Contexts of One Trillion Words for WSD
Tobias Hawker 85

Word Sense Disambiguation in Contextual Dynamic Network Using Associative Concept Dictionary
Jun Okamoto and Shun Ishizaki 94

Large-Scale Supervised Models for Noun Phrase Bracketing
David Vadas and James Curran 104

Empirical study on Parsing Chinese Based on Collins Model
Hailong Cao, Yujie Zhang and Hitoshi Isahara 113

Learning-Based Argument Structure Analysis of Event-Nouns in Japanese
Mamoru Komachi, Ryu Iida, Kentaro Inui and Yuji Matsumoto 120

Interpreting Noun Compounds using Bootstrapping and Sense Collocation
Su Nam Kim and Timothy Baldwin 129

Semantic Analysis of Entity Contexts towards Open Named Entity Classification on the Web
Xuan-Hieu Phan, Le-Minh Nguyen, Cam-Tu Nguyen and Susumu Horiguchi 137

Recognizing Names by the Meaning of Defining Syntactic Structures
Li Yang and Steven Abney 145

A FrameNet Approach to Connectives: The Polysemy of While
Satoru Uchida and Seiko Fujii 154

Resolution of Coordination Ellipses in Biological Named Entities with Conditional Random Fields
Ekaterina Buyko, Katrin Tomanek and Udo Hahn 163

Minimising semantic drift with Mutual Exclusion Bootstrapping
James Curran, Tara Murphy and Bernhard Scholz 172

Analogical Mapping Method and Semantic Categorization of Japanese Compound and Complex Sentence Patterns
Satoru Ikehara, Masato Tokuhisa and Jin'ichi Murakami 181

Bridging the gap: Thai - Thai Sign Machine Translation
Srisavakon Dangsaart, Kanlaya Naruedomkul, Nick Cercone and Booncharoen Sirinaovakul . 191

Preparing, Restructuring, and Augmenting a French Treebank: Lexicalised Parsers or Coherent Treebanks?
Natalie Schluter and Josef van Genabith 200

Improving the complement/adjunct distinction in CCGbank
Matthew Honnibal and James Curran 210

Evaluating Feature Types for Encoding Clinical Notes
Jon Patrick, Yitao Zhang and Yefeng Wang 218

Global Revision in Summarisation: Generating Novel Sentences with Prim's Algorithm
Stephen Wan, Robert Dale, Mark Dras and Cécile Paris 226

Modelling and Exploring the Network Structure of Terminology by Potts Spin Glass Model
Kyo Kageura and Takeshi Abekawa 236

Determining the Unithood of Word Sequences: Mutual Information and Independence Measure
Wilson Wong, Wei Liu and Mohammed Bennamoun 246

A case study of comparison of several methods for corpus-based speech intention identification
Kazutaka Shimada, Kaoru Iwashita and Tsutomu Endo 255

Author Profiling for English Emails
Dominique Estival, Tanja Gaustad, Ben Hutchinson, Son Bao Pham and Will Radford 263

Towards a Semantic Analysis of Bahasa Indonesia for Question Answering
Septina Dian Larasati and Ruli Manurung 273

Query Recommendation from Search Engine Query Logs Using Query Enrichment
Shisanu Tongchim, Virach Sornlertlamvanich and Hitoshi Isahara 281

Automagically Inferring the Source Words of Lexical Blends
Paul Cook and Suzanne Stevenson 289

Discovering Word Senses for Polysemous Words Using Feature Domain Similarity
Noriko Tomuro, Kyoko Kanzaki and Hitoshi Isahara 298

Building a clustered semantic network for an Entire Large Dictionary of Japanese
Hiroyuki Akama, Jaeyoung Jung, Maki Miyake, Kenji Yosihasi and Kikuko Nishina 308

A Named Entity Recogniser for Question Answering
Menno van Zaanen and Diego Mollá 317

Effect of combining different Web search engines on Web question-answering
Tatsunori Mori, Akira Kanai, Madoka Ishioroshi and Mitsuru Sato 325

What to Classify and How: Experiments in question classification for Japanese
Rebecca Dridan and Timothy Baldwin 333

Sentence retrieval for extracting biomedical knowledge
Tara McIntosh and James Curran 342

Centering in Chinese Anaphor Resolution: A Parametric Study
Manjuan Duan 350

Conference Program

Wednesday, September 19, 2007

9:45–10:00 **Opening Remarks (Th 3)**

	Session 1a: Querying (Th 3)	**Session 1b: Lexical Semantics (Th 2.06)**
10:00–10:30	*Pinyomi: Dictionary lookup via orthographic associations* Lars Yencken, Zhihui Jin and Kumiko Tanaka-Ishii	*Detecting Compositionality of English Verb-Particle Constructions using Semantic Similarity* Su Nam Kim and Timothy Baldwin
10:30–11:00	*Corpus Statistics Approaches to Discriminating Among Near-Synonyms* Mary Gardiner and Mark Dras	*Graphical query for linguistic treebanks* Steven Bird and Haejoong Lee

11:00–11:30 **Coffee break**

Invited Talk (Th 3)

11:30–12:45 *The Generation of Referring Expressions: Where We've Been, How We Got Here, and Where We're Going*
Invited Talk by Robert Dale

12:45–14:00 **Lunch**

	Session 2a: Segmentation (Th 3)	**Session 2b: WSD (Th 2.06)**
14:00–14:30	*Sentence and Token Splitting Based On Conditional Random Fields* Katrin Tomanek, Joachim Wermter and Udo Hahn	*Combining Similarity-based Approaches for Semantic Tagging of Unknown Words* Fumiyo Fukumoto and Yoshimi Suzuki
14:30–15:00	*Using Conditional Random Fields for Clause Splitting* Vinh Van Nguyen, Minh Le Nguyen and Akira Shimazu	*Using Contexts of One Trillion Words for WSD* Tobias Hawker
15:00–15:30	*Incorporating Global Information into Supervised Learning for Chinese Word Segmentation* Hai Zhao and Chunyu Kit	*Word Sense Disambiguation in Contextual Dynamic Network Using Associative Concept Dictionary* Jun Okamoto and Shun Ishizaki

15:30–16:00 **Coffee break**

	Session 3a: Parsing (Th 3)	**Session 3b: Lexical Semantics (Th 2.06)**
16:00–16:30	*Large-Scale Supervised Models for Noun Phrase Bracketing* David Vadas and James R. Curran	*Learning-Based Argument Structure Analysis of Event-Nouns in Japanese* Mamoru Komachi, Ryu Iida, Kentaro Inui and Yuji Matsumoto
16:30–17:00	*Empirical study on Parsing Chinese Based on Collins' Model* Hailong Cao, Yujie Zhang and Hitoshi Isahara	*Interpreting Noun Compounds using Bootstrapping and Sense Collocation* Su Nam Kim and Timothy Baldwin

RECEPTION (17:30–19:00)

Thursday, September 20, 2007

	Session 4a: Named Entities (Th 3)	**Session 4b: Coordination/Conjunction (Th 2.06)**
10:00–10:30	*Semantic Analysis of Entity Contexts towards Open Named Entity Classification on the Web* Xuan-Hieu Phan, Le-Minh Nguyen, Cam-Tu Nguyen and Susumu Horiguchi	*A FrameNet Approach to Connectives: The Polysemy of While* Satoru Uchida and Seiko Fujii
10:30–11:00	*Recognizing Names by the Meaning of Defining Syntactic Structures* Li Yang and Steven Abney	*Resolution of Coordination Ellipses in Biological Named Entities with Conditional Random Fields* Ekaterina Buyko, Katrin Tomanek and Udo Hahn
11:00–11:30	**Coffee break**	
	Invited Talk (Th 3)	
11:30–12:45	*Applying robust semantics* Invited Talk by Ann Copestake	
12:45–14:00	**Lunch**	
	Session 5: Lexical Semantics (Th 3)	
14:00–14:30	*Minimising semantic drift with Mutual Exclusion Bootstrapping* James Curran, Tara Murphy and Bernhard Scholz	
	Session 6a: Machine Translation (Th 3)	**Session 6b: Parsing (Th 2.06)**
14:30–15:00	*Analogical Mapping Method and Semantic Categorization of Japanese Compound and Complex Sentence Patterns* Satoru Ikehara, Masato Tokuhisa and Jin'ichi Murakami	*Preparing, Restructuring, and Augmenting a French Treebank: Lexicalised Parsers or Coherent Treebanks?* Natalie Schluter and Josef van Genabith
15:00–15:30	*Bridging the gap: Thai - Thai Sign Machine Translation* Srisavakon Dangsaart, Kanlaya Naruedomkul, Nick Cercone and Booncharoen Sirinaovakul	*Improving the complement/adjunct distinction in CCGbank* Matthew Honnibal and James Curran
15:30–16:00	**Coffee break**	
	Session 7a: Terminology (Th 3)	**Session 7b: Meta Information (Th 2.06)**
16:00–16:30	*Modelling and Exploring the Network Structure of Terminology by Potts Spin Glass Model* Kyo Kageura and Takeshi Abekawa	*Evaluating Feature Types for Encoding Clinical Notes* Jon Patrick, Yitao Zhang and Yefeng Wang
16:30–17:00	*Determining the Unithood of Word Sequences: Mutual Information and Independence Measure* Wilson Wong, Wei Liu and Mohammed Bennamoun	*Global Revision in Summarisation: Generating Novel Sentences with Prim's Algorithm* Stephen Wan, Robert Dale, Mark Dras and Cecile Paris

BANQUET (19:00–21:00)

Friday, September 21, 2007

	Session 8a: Meta Information (Th 3)	**Session 8b: Question Answering (Th 2.06)**
10:00–10:30	*A case study of comparison of several methods for corpus-based speech intention identification* Kazutaka Shimada, Kaoru Iwashita and Tsutomu Endo	*Towards a Semantic Analysis of Bahasa Indonesia for Question Answering* Septina Dian Larasati and Ruli Manurung
10:30–11:00	*Author Profiling for English Emails* Dominique Estival, Tanja Gaustad, Ben Hutchinson, Son Bao Pham and Will Radford	*Query Recommendation from Search Engine Query Logs Using Query Enrichment* Shisanu Tongchim, Virach Sornlertlamvanich and Hitoshi Isahara
11:00–11:30	**Coffee break**	
	Invited Talk (Th 3)	
11:30–12:45	*Word Construction and Multi-layered Sentence Analysis* Invited Talk by Yuji Matsumoto	
12:45–14:00	**Lunch**	
	Session 9a: Lexical Semantics (Th 3)	**Session 9b: Question Answering (Th 2.06)**
14:00–14:30	*Automagically Inferring the Source Words of Lexical Blends* Paul Cook and Suzanne Stevenson	*A Named Entity Recogniser for Question Answering* Menno van Zaanen and Diego Mollá
14:30–15:00	*Discovering Word Senses for Polysemous Words Using Feature Domain Similarity* Noriko Tomuro, Kyoko Kanzaki and Hitoshi Isahara	*Effect of combining different Web search engines on Web question-answering* Tatsunori Mori, Akira Kanai, Madoka Ishioroshi and Mitsuru Sato
15:00–15:30	*Building a clustered semantic network for an Entire Large Dictionary of Japanese* Hiroyuki Akama, Jaeyoung Jung, Maki Miyake, Kenji Yosihasi and Kikuko Nishina	*What to Classify and How: Experiments in question classification for Japanese* Rebecca Dridan and Timothy Baldwin
15:30–16:00	**Coffee break**	
	Session 10a: Information Extraction (Th 3)	**Session 10b: Anaphor Resolution (Th 2.06)**
16:00–16:30	*Sentence retrieval for extracting biomedical knowledge* Tara McIntosh and James Curran	*Centering in Chinese Anaphor Resolution: A Parametric Study* Manjuan Duan
	Special talks (Th 3)	
16:30–17:00	*An Introduction to Research at NICT*	
17:00–17:30	*An Introduction to Research at NICTA*	
17:30–17:45	**Closing Remarks (Th 3)**	

Applying Robust Semantics

Ann Copestake
Computer Laboratory, University of Cambridge
JJ Thomson Avenue, Cambridge, UK
aac@cl.cam.ac.uk

Abstract

Semantic representations provide an abstraction which allows generalisations to be made across very different surface forms and different syntactic formalisms. But deriving conventional semantic representations requires more information than can be extracted from morphology and syntax, even by deep grammars. The standard way of dealing with this is to use underspecified semantic representations, which can, in principle, be monotonically enriched to provide deeper semantics with which inference engines can work. However, an alternative is to operate directly on the underspecified representations. In this paper, I will discuss 'matching', 'merging' and 'munging' operations in the context of the (Robust) Minimal Recursion Semantics approach and give an overview of some of the uses to which they have been put.

1 Introduction

In the 1980s, the paradigm example of an application of natural language processing was the natural language interface. The idea was that a user could communicate with a system by typing text rather than entering a formal command language. For instance, natural language interfaces were built that could access databases in order to answer queries such as:

(1) Who had the highest sales figures in June 1982?

(2) Is there a doctor on board the Vincennes?

Although several of the later systems used a general purpose core grammar, all the systems relied heavily on the use of domain-specific lexicons and on exploiting constraints that came from the underlying database to resolve ambiguity. For instance, in (1), the term *sales figures* would be directly encoded in the lexicon, and would map directly or indirectly to a database concept. Such systems thus had precise notions of denotation with respect to a database and it was viable to exploit various forms of inference. Much of the most important early work on computational compositional semantics was done in the context of such applications. However, the systems had limited commercial success (for a variety of reasons, not all to do with the limitations of the natural language interfaces).

Most researchers are now more interested in applications that involve broad-coverage text processing, especially applications designed to help users get information out of some large collection of text. At one extreme, web search returns full documents and typically involves little language processing. Information extraction (IE) operates on relatively unrestricted text to acquire specific types of information that can be used to instantiate fixed templates. Question answering (QA), like IE, is designed to provide the user with targeted information, but the topics of the questions are not known in advance. QA involves matching the user query with pieces of text in a document collection and returning short text snippets or precise answers. Machine translation and summarisation can perhaps also best be seen as facilitating information gathering operations of various types.

Until relatively recently, there was little attempt to use compositional semantics in broad-coverage applications. However, this has changed, for a number of reasons. At the technological level, it has become possible to parse large quantities of text with relatively deep parsers that can support forms of semantic composition. Statistical techniques for parse ranking are effective on deep grammars and can be used to disambiguate even when there are millions of analyses. Entailment has been recognised as a common component for a range of applications and robust entailment is becoming a recognised notion, via the shared Recognising Textual Entailment (RTE) task (Dagan et al., 2006). The AQUAINT project in the US has funded a range of research into robust seman-

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 1–12,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

tics. Finally, as the information available online continues to increase, users demand more precise results than conventional search can provide.

Broad-coverage/large-scale text processing is fundamentally different from research on domain-dependent systems. Firstly, some semantic distinctions cannot be made without a domain model. Detailed lexicons are difficult and expensive to obtain. The early domain-dependent systems sometimes used very detailed hand-crafted lexicons but these were small. There are no generally available broad-coverage lexicons with good information about multiword expressions, aspectual classes of verbs, nouns which are likely to appear as mass terms, nouns which appear in pseudo-partitive constructions (such as *loads of*) and so on. Secondly, most natural language interface systems assumed that queries were isolated utterances. However, for applications operating with large text collections, it is essential to allow for inter-sentential anaphora resolution. Although this is not a requirement for formal semantic representation in linguistics, the only alternative in a computational approach is for anaphora resolution to work off the syntactic representation. This is unattractive, especially since in many cases some inference is required to select between antecedents or to construct possible antecedents from conjuncts. Semantic representations must also be compatible with representations of text structure. Finally, multiplication of readings in semantic interpretation must be avoided as far as possible. This leads to devices for representing underspecified information, discussed in more detail below.

For the earlier work on natural language interfaces, it was to some extent viable to adopt the approach favoured in much of the literature on formal semantics, where the logical form itself is seen as merely a convenience for getting at the underlying denotation. For computational work on interfaces to databases, the denotation can be treated as the contents of the database (although interpreting questions under this assumption involves some fudging). This is clearly not possible in broad-coverage work, where the semantic representation itself is important and has to be suitable for supporting tasks such as matching and anaphora resolution. In underspecified semantic representations, the models are often seen as fully scoped logical forms.

Some researchers use automatically extracted semantic representations from broad-coverage systems with theorem provers (e.g., Moldovan et al. (2007) and Bos and Markert (2005)) in conjunction with robust methods. Moldovan et al's COGEX prover incorporates various techniques for relaxing proofs. Strictly applied theorem proving is unsuitable for broad-coverage semantics for a range of reasons, but most importantly because it is not robust to missing information and only very limited broad-coverage knowledge sources are available. The work discussed in this paper does not use theorem proving but treats semantic representations as a way of providing a useful level of abstraction for forms of matching operations and rewriting.

The aim of this paper is to discuss work on broad-coverage computational semantics derived from deep grammars from an applications viewpoint. Relevant work on broad coverage semantics includes van Genabith and Crouch (1999); Baldridge and Kruijff (2002); Gardent and Kallmeyer (2003); Ge and Mooney (2005); Bos (2005). However here I will concentrate on discussing work which has used Minimal Recursion Semantics (MRS: Copestake et al. 2005 or Robust Minimal Recursion Semantics (RMRS: Copestake 2003). The MRS/RMRS approach has been adopted as a common framework for the DELPH-IN initiative (Deep Linguistic Processing with HPSG: `http://www.delph-in.net`) and the work discussed here has mostly been done by researchers connected to DELPH-IN.

I will start by discussing the general DELPH-IN approach to semantic representation in §2 and introduce the MRS and RMRS languages. We have found that we can use semantic representations from broad-coverage, domain-independent grammars in a variety of ways, using a set of core operations on the semantic output. In §3, I give some definitions of matching, merging and munging (mapping) operations on RMRSs (the MRS operations are similar, so it would be redundant to discuss both). Although various implementations of these operations have been around since the mid-1990s, their use has not been described in detail previously. In §4, I illustrate the use of these operations in applications, and briefly mention some of the projects which have used (R)MRS.

The mixture was allowed to warm to room temperature.

⟨ l3:a1:_the_q(x5)⟨0,3⟩, l7:a2:_mixture_n(x5)⟨4,11⟩, l9:a3:_allow_v_1(e2)⟨16,23⟩,
l13:a5:_warm_v_1(e14)⟨27,31⟩, l13:a6:_to_p(e15)⟨32,34⟩, l17:a7:udef_q(x16)⟨35,52⟩,
l20:a8:compound(e22)⟨35,52⟩, l23:a9:udef_q(x21)⟨35,52⟩, l26:a10:_room_n(x21)⟨35,39⟩,
l20:a11:_temperature_n(x16)⟨40,52⟩⟩
⟨ a1:RSTR(h6), a1:BODY(h4), a3:ARG2(x5), a3:ARG3(h10), a5:ARG1(x5), a6:ARG1(e14),
a6:ARG2(x16), a7:RSTR(h18), a7:BODY(h19), a8:ARG1(x16), a8:ARG2(x21), a9:RSTR(h24),
a9:BODY(h25) ⟩
⟨ qeq(h6,l7), qeq(h18,l20), qeq(h24,l26) ⟩

RMRS structure

⟨ l3:_the_q(x5,h6,h4), l7:_mixture_n(x5), l9:_allow_v_1(e2,x5,h10), l13:_warm_v_1(e14,x5),
l13:_to_p(e15,e14,x16), l17:udef_q(x16,h18,h19), l20:compound(e22,x16,x21),
l23:udef_q(x21,h24,h25), l26:_room_n(x21), l20:_temperature_n(x16) ⟩
⟨ qeq(h6,l7), qeq(h18,l20), qeq(h24,l26) ⟩

MRS structure

_the_q(x5,_mixture_n(x5),_allow_v_1(e2,u11,x5, udef(x21, _room_n((x21),
udef(x16,compound(e22,x16,x21) ∧ _temperature_n((x16), warm_v_1(e14,x5) ∧ _to_p(e15,e14,x16)))))

One of the scoped forms corresponding to a specialisation of the MRS above.

Figure 1: Semantic representations

2 Flat semantics

The goal of computational compositional semantics is to represent the information we can derive from the syntax and inflectional morphology correctly in a way that facilitates further processing. It is essential that the compositional semantic representation only includes information that is known, as opposed to multiplying structures. This has led to a style of semantic representation which is very different from that found in linguistics textbooks. To illustrate this, see the example in Figure 1. The RMRS for the preferred parse for the sentence and the MRS are shown. One of the scoped logical forms is at the bottom of the figure. These structures are output from the English Resource Grammar (ERG: Flickinger 2000, version LinGO Jul-07). Some aspects have been simplified: in particular, sortal information on variables, which encodes tense, aspect, plurality and so on, is not shown.

RMRSs can be output both by parsers running deep grammars (such as the ERG) and by shallower processing. The first element in the RMRS structure is a list of elementary predications (EPs). Most of these correspond to lexemes in the input (the predicates in the EP start with an underscore in these cases). The naming convention adopted for the relations (e.g., _temperature_n) allows them to be constructed without access to a lexicon (which is required for shallower processing and for unknown word handling for deep processing). '_n', '_v' etc are indications of the coarse-grained sense distinctions which can be inferred from part-of-speech information. Deep grammars with relatively detailed lexicons may produce finer-grained sense distinctions, indicated by '_1' etc, and there is an implicit hierarchy such that _allow_v_1 is more specific than _allow_v. Besides the relation, an EP has a label (e.g., l3) and anchor (e.g., a1) and at most one argument (the ARG0, or characteristic argument of the EP). In the figure, each EP is also decorated with a record of the character positions it corresponds to in the input. Both MRS and RMRS achieve a flat representation via the use of labels on EPs, thus factoring out scopal relationships. Implicit conjunction is indicated by equality between labels. In Figure 1, the EP corresponding to *warm* and the one corresponding to the preposition *to* share a label.

The second component in the RMRS is the list of argument relations. In RMRS, arguments (ARGs) are represented as distinct elements (a 'neo-Davidsonian' representation). The ARGs are related to the main EPs via the anchor element. An EP and its associated ARGs share a unique

anchor.[1] Most arguments in the figure are 'ordinary' variables (nominal objects, indicated by x, or events, e), but the ARG3 of *allow* and the RSTR and BODY arguments of the quantifiers are 'hole' arguments, which indicate scopal relationships.

The MRS representation differs from RMRS primarily in being less factored, as the arguments are part of the EPs. The RMRS representation may be output from grammars which do not have a lexicon and where the arguments are derived from the construction, unlike MRS and most other semantic representation languages, where the arity of predicates has to be lexically specified. An RMRS may omit argument relations and be well-formed, although underspecified. The ARG relations themselves may also be underspecified: e.g., ARGn indicates that there is some argument relationship, but it is unknown whether it is an ARG1, ARG2 or ARG3. In fact, a bag of lexeme-derived EPs with no relational information at all could be a valid RMRS and this can be constructed directly from a POS tagged sequence. Thus deep and shallow analysers can output compatible representations, at least up to a point.

The third component in the RMRS (second in the MRS) constrains the scopes that may be assigned to a structure. Here the constraints are qeq relationships (equality modulo quantifiers). Hole arguments are always qeq an EP label, which indicates that the EP either instantiates that argument position directly, or that one or more quantifiers intervene. These constraints are discussed in Copestake et al. (2005) but the details are not important to understand this paper. The final structure in Figure 1 shows one of the scoped forms compatible with the qeq constraints.

The udef and compound relations come from the phrase *room temperature* which is analysed syntactically as a noun-noun compound forming a noun phrase with no determiner (a bare NP). Semantically, the ERG treats both the nouns as having an underspecified quantifier. The compound relation in (R)MRS is also to be understood as very underspecified. Discussion of the semantics of English nominal compounds would be far too involved for the current paper, but note that this is a case where a very simple representation is preferred over an attempt to provide an analysis of the precise nature of the compound, which cannot be determined by the grammar. If it is necessary to try and determine the compound relation more precisely for a particular application, this can be viewed as specialising the compound relation in the original (R)MRS. The output (R)MRS contains all the information derivable from syntactic parsing which might be useful for data-driven techniques for compound relation determination, so this can be done after parsing.

The explicit record of the original character positions in the RMRS is clearly not part of the 'real' semantics but it has a number of uses, including facilitating anaphora resolution and allowing default quantifier scope readings on the basis of surface order. For some of these uses, it is a stopgap measure in the absence of a suitable underspecifiable representation for information structure. However it is also important for the tractability of the merging operation discussed in §3.3, below.

The representation of compositional semantics using RMRS can be seen as augmenting the information that comes from a simple lexeme sequence. The motivation for the flat semantics style is essentially practical, since it allows for underspecification and a very simple notion of composition (see Copestake (2007), but not discussed here) and also facilitates generation and other robust operations on the semantics, as discussed below. But there is also a more theoretical motivation: since natural languages are the only successful general knowledge representations that exist, we might be well-advised to see semantics as (partially) specifying and disambiguating a natural language string, rather than as replacing it completely with an alternative.[2] In general, the techniques in formal semantics which have been widely adopted within computational linguistics have tended to be those which facilitate a relatively 'surfacy' notation. This includes the use of generalized quantifiers, event variables, the underspecification of compound relations and Hobbs' 'ontological promiscuity' idea (Hobbs, 1985). In some cases, computational linguists over-simplify and ignore semantically meaningful distinctions which could be derived, but generally the motivation is

[1] This is a change from Copestake (2003): the reasons for this are discussed in Copestake (2007).

[2] There is some relationship between this idea and the 'Natural Logic' approach. See MacCartney and Manning (2007) for what is apparently the first broad-coverage system using a natural logic approach. However, Natural Logic is generally proposed as a way of supporting (putatively) human-like reasoning, while our interest in using a very 'surfacy' representation primarily stems from a requirement for robustness in composition.

to keep unresolvable distinctions out of the initial semantic representation while allowing those distinctions to be made by later enrichment.

3 Operations on RMRS

In this section, I give brief and rather informal specifications of some of the operations on RMRS representations. Applications of these are discussed in the next section. The discussion is centred on RMRS because MRS operations are similar but simpler. This description is an overview of RMRS operations rather than a full account: there are a number of implementations (developed by Rebecca Dridan, Stephan Oepen and Ulrich Callmeier as well as by myself) and these have variants not discussed here.

3.1 The RMRS language

An RMRS EP contains:

1. the label of the EP: this is shared by other EPs to indicate implicit conjunction.
2. an anchor, not shared by any other EPs.
3. a relation
4. up to one argument of the relation (the arg0)

This is written as label:anchor:relation(arg0).

An RMRS ARG relation contains:

1. an anchor, which must also be the anchor of an EP.
2. an ARG relation, taken from a fixed set (here: ARG1, ARG2, ARG3, RSTR, BODY, plus the underspecified relations: ARG1-2, ARG1-3, ARG1-2, ARG2-3, ARGN).
3. exactly one argument. This must be 'grounded' by an EP: i.e., if it is a normal variable it must be the ARG0 of an EP, or if it is a hole, it must be related to the label of an EP by a qeq constraint.

An RMRS structure contains:

1. rels: The bag of EPs.
2. args: The bag of argument relations.
3. hcons: qeq constraints. A qeq relationship always holds between a hole in an argument relation and the label of an EP.

3.2 Matching

The basic matching operation checks for consistency between two RMRSs rather than identity or implication. The intuitive idea of matching and merging is closely related to the concept of feature structure unification: two RMRSs can be said to match iff there is a more specific RMRS which is subsumed by both of them. The most general such subsumed RMRS is the merge of the two RMRSs.

However, RMRSs can only be merged to give a unique result if an external ordering is available on EPs, as discussed below, and if the ordering on predicates and argument relations forms a bounded complete partial order (BCPO). These conditions can be enforced when RMRSs derived from the same string are merged, but will not be true in general. Furthermore, we require a definition of matching that allows for partial matches. Hence the description below takes EP matching as primary.

An EP1 matches EP2 if:

1. the relation associated with EP1 is compatible with the relation associated with EP2. The notion of compatibility may vary according to the application, but in general the assumption is that there is a partial order on relations.
2. the arg0 associated with EP1 is compatible with the arg0 associated with EP2. This includes a check on the values encoding tense and aspect.
3. Neither of the anchors are already matched.

If EP1 matches EP2, there are the following variable equivalences: $l_1/l_2, a_1/a_2, arg0_1/arg0_2$. The full set of variable equivalences arising from matching two RMRSs is notated RMRS1/RMRS2.

Matching argument relations depends on RMRS1/RMRS2. ARG-REL1 matches ARG-REL2 iff:

1. The anchor of ARG-REL1 is bound to the anchor of ARG-REL2 in RMRS1/RMRS2.
2. and, if the argument of ARG-REL1 is a normal variable, it is bound to the argument of ARG-REL2 in RMRS1/RMRS2.
3. or, if the argument of ARG-REL1 is a hole, it is qeq a label which is bound to a label l2

in RMRS1/RMRS2 such that the argument of ARG-REL2 is qeq l2.

4. and the relation in ARG-REL1 is compatible with the relation in ARG-REL2 (i.e., they have a greatest lower bound in the partial order of argument relation types).

If the argument relations match under this definition, the qeq relationships must also match. Hence two RMRSs R1 and R2 match iff:

- each EP in R1 matches an EP in R2
- each EP in R2 matches an EP in R1
- each argument relation in R1 matches an argument relation in R2
- each argument relation in R2 matches an argument relation in R1

Note that the ordering of EPs and argument relations is not significant.

There are a range of variants on this notion of matching which are useful in different circumstances:

- RMRSs may be checked for subsumption rather than compatibility. This is useful in the definition of idiom patterns (§4.2).
- RMRS patterns may be used rather than matching two RMRSs derived by a grammar. This is useful for IE.
- Unmatched EPs may be allowed on one or both sides (used in QA).
- A robust weighted match can be defined which is given a score according to which EPs match and whether their arguments match. This is also used in QA.
- A match may signal some action: this is used to postulate 'null semantic items' for generation (see below).

3.3 Merging

Merging two semantic structures can be regarded as a semantic conjunction operation. Merging may be used when RMRSs are constructed from more than one analyser. The idea is that if a deep analyser fails to find a complete analysis, it may be patched up with output from a shallower analyser. The Heart of Gold machinery developed for the Deep Thought project implements this sort of idea (Callmeier et al., 2004). Merging can also be used to produce a packed RMRS representation: structures which are not fully compatible can be merged to create a lattice.

The definition of merging argument relations depends on matching the EPs. If this were not the case, the result of merging could be ambiguous, due to the possibility of underspecification of argument relations. For instance, suppose we attempt to merge $l : a : P(i)$, $a : \mathrm{ARGN}(x)$, $a : \mathrm{ARGN}(y)$ with $l' : a' : P'(i')$, $a' : \mathrm{ARGN}(x')$, $a' : \mathrm{ARGN}(y')$ where P matches P'. There are two possible matches: in one x matches x' and y matches y' while in the other x matches y' and y matches x'. However, an RMRS is only well-formed if all argument relations have variables grounded in EPs: i.e., there must be some EP $l_a : a_a : P_a(x)$ which grounds x and similarly $l_b : a_b : P_b(x')$, $l_c : a_c : P_c(y)$ and $l_d : a_d : P_d(y')$. EPs are ordered by character position and, in the case of those construction EPs which come from the same phrase where character position does not discriminate, they are ordered according to position in the derivation tree (e.g., in Figure 1, the two udef relations are ordered because the compound-noun rule forms a phrase which is the daughter in the bare noun rule). Thus the result of the merge is unique.

3.4 Munging

The original 'munging' rules were an expedient to allow MRSs output by earlier versions of the ERG to be converted into standardised representations. The basic idea is to have rules consisting of three components: an input, an output and a context, all of which are optional. The interpretation is that if the input of the rule matches part of an MRS, the matching portion of the MRS is transformed to the output. The optional context allows for conditions to be set which are unaffected by the mapping. A rule with a context but no input specification causes insertion, a rule with no output specification causes deletion. Each rule can be applied multiple times to a given MRS. Rulesets are ordered, and rules are applied in sequence, but there is no reapplication of rules. A refined version of this approach was developed by Stephan Oepen and used for semantic transfer for machine translation on the LOGON project (see e.g., Flickinger et al. 2005). It can also be used for monolingual paraphrase and for mapping input to a domain-

specific representation.

4 Applications of robust operations

The applications discussed here all make use of some form of the matching operation. The purpose of this section is not just to illustrate that the operation is practically useful, but to give some indication of the way its use relates to more conventional semantic operations.

4.1 Semantic operations in generation

The ERG and other DELPH-IN grammars are bidirectional: they can be used to generate strings from (R)MRSs as well as to analyse them. Indeed the design of the MRS language was heavily influenced by the need to make it usable as the input to the generator.

The idealisation of generating from an input meaning representation is that all and only the strings which have 'compatible' semantics to the input should be produced. If 'compatible' is interpreted as 'logically equivalent', then if LF1 is constructed by grammar G from string S, and LF2 is logically equivalent to LF1, then a generator working with grammar G should accept LF2 and output string S. Unfortunately this notion of compatibility leads to undecidability even for first order predicate calculus (Shieber, 1993).

However, it is not clear that logical equivalence gives intuitively useful results. Suppose the input is the following fully scoped logical form:

$$\begin{aligned}&\text{not}(\text{every}(x, \text{black}(x) \wedge \text{dog}(x),\\ &\text{every}(y, \text{cat}(y), \text{likes}(x, y))))\end{aligned}$$

Sentences which might be generated if logical equivalence is taken as the criterion include:

(3) It is not the case that every black dog likes every cat.

(4) There's some black dog that does not like some cat.

Although these sentences are logically equivalent taken in isolation, they are unlikely to both be equally appropriate in a particular discourse. Furthermore, quantifiers such as *every* are quite rare in most texts. So the sort of logical manipulation which involves converting universals to existential and introducing negation is not generally important in practical applications of generation. We decided instead to investigate a more limited notion of equivalence, which is essentially that given by the (R)MRS matching operation described above. For example, consider the following scoped representation:

$$\begin{aligned}&\text{this}(x, \text{path}(x) \wedge \text{steep}(x) \wedge \text{slippery}(x),\\ &\text{the}(y, \text{stream}(y), \text{follow}(x, y))))\end{aligned}$$

The sentences which can be generated if (R)MRS matching is taken as the criterion include:

(5) This steep slippery path follows the stream.

(6) This slippery steep path follows the stream.

(7) The stream is followed by this steep slippery path.

This notion of equivalence has been assumed for the chart generation approach used with the DELPH-IN grammars (Carroll et al., 1999; Carroll and Oepen, 2005). The algorithm depends on the monotonicity constraint on composition with (R)MRS: the EPs in a phrase are the EPs of the daughters of the phrase, plus any contribution from the construction. No EPs may be deleted during composition. Generation operates in two main phases: in the first, lexical entries are collected on the basis of the input semantics and used to instantiate a chart. The tractability of this depends on being able to split up the (R)MRS into subsets of EPs which correspond to lexical entries. In the second phase, grammar rules are applied to these entries, much as in parsing. The algorithm keeps track of the input EPs it has accounted for, which again is a process which the flat structure considerably facilitates. The (R)MRSs associated with the different realisations are checked for compatibility with the original input (R)MRS at the end of the process.

The first stage, of identifying lexical entries which are candidates for inclusion in the initial chart, has to allow for lexical entries which do not have associated relations (null semantics): e.g., infinitival *to*, expletive *there*. These are introduced when licensed by 'trigger' rules which pair a context with a lexical identifier. When the context matches the input (R)MRS, a lexical entry corresponding to the identifier is added to the chart. For example, the following (trivial) rule licences the introduction of the lexical entry for expletive *there* just in case a particular subclass of *be* relation is found in the input.

```
l:a:_be_v_there_rel(e) -> "there_expl"
```

4.2 Idioms

The concept of (R)MRS matching is also used in an approach to representing idioms. Most idioms are 'compositional' in the sense that the meaning of the idiomatic phrase can be treated as composed of the meaning of the component parts, even though the words concerned do not have their usual senses. Examples include *take heart*, *spill beans*, *cat out of (the) bag*. For instance, *spill the beans* corresponds roughly to *reveal the secrets*. Syntactic variation is possible for these idioms, as is modification. For instance, (8) is found in the British National Corpus.

(8) We take considerable heart from the knowledge ...

In the approach used in the ERG, the lexicon includes idiomatic senses for words such as *heart*. These are normal lexical entries in most respects (although possibly with different syntactic behaviour from other senses of the word in question), but their use is constrained so that they can only appear in sentences along with the other components of the idiom. This constraint is expressed in terms of a semantic pattern which is treated as a root condition: i.e., as a constraint which must be obeyed by the semantics of the utterance as a whole. For instance, the following RMRS pattern specifies that the idiomatic sense of *heart* must be the ARG2 of the idiomatic *take* (idiomatic senses are indicated by '_i'):

```
l:a:_take_v_i(e), ARG2(a,x),
l1:a1:_heart_n_i(x)
```

The constraint applies to any sentence where either idiomatic sense is used. This allows for examples such as (8) as well as the unmodified cases. The approach is discussed in Riehemann (2001). Pulman (1993) describes a somewhat similar approach which is also based on the use of semantic patterns, although in Pulman's case these operate on the literal interpretation.

4.3 Search, IE and QA

Robust matching on RMRSs has been applied to IE in the Deep Thought project (Callmeier et al., 2004) and is currently being applied to Chemistry texts on the SciBorg project (Copestake et al., 2006). (R)MRS has also been used in the extraction of ontologies from Wikipedia (Herbelot and Copestake, 2006) and from dictionaries (Nichols et al., 2005) (both can be regarded as a form of IE). Work on QA using RMRS has been done in a series of student experiments at Cambridge[3], by Watson et al. (2003), and also by Dridan (Dridan, 2006; Dridan and Bond, 2006). The QUETAL project (Frank et al., 2007) has also used RMRS for QA from structured knowledge resources. For concreteness, I will discuss general QA in this section, although much of the discussion applies to search and IE as well.

The basis for the use of RMRS in QA is matching between the RMRS corresponding to a query (with the representation for wh-words omitted) and a subpart of an answer RMRS. This thus takes into account more context than word overlap does. For instance, in the query (9), using RMRS as opposed to simple word overlap avoids matches on texts about what jellyfish eat as opposed to what eats them.[4]

(9) What eats jellyfish?

The idea that an answer should entail a query (with suitable bindings for the queried terms) is well known. If an RMRS representation of a query fully matches part of a sentence, then whether there is a genuine entailment will depend on the remainder of the sentence. If the unmatched part of the context only involves intersective modification, conjunction and so on, then the putative answer will logically entail the query. If the matched RMRS is inside a (single) negation, then the entailment will not hold. In modal contexts, and in sentential complements to verbs of belief, assertion and so on, an entailment which is not valid logically might nevertheless be taken as valid for the RTE task. If all the EPs in an RMRS were conjoined, then a match between some of the EPs and a query would result in a logically valid entailment. In effect, robust matching using RMRS amounts to treating the bag of EPs as though they were conjoined, but because the extra information about the scopal relationships is available in

[3]These experiments have been done as part of the language practicals in the Cambridge Masters course in Computer Speech Text and Internet Technology. The practicals are run by Ted Briscoe, Simone Teufel and the author.

[4]Of course, we also need to consider cases where the terminology in the texts differs from that of the query. RMRS munging rules can be used for query reformulation, but then the main issue is how to obtain the rules in the first place, which I will not discuss further here since the same problem applies to other approaches too.

the RMRS, the simple notion of matching that ignores the surrounding EPs can be refined to avoid some incorrect entailments. This distinguishes RMRS approach from simpler accounts of semantics which do not allow for scope.

To illustrate this, consider the example query (9) again. For all of the candidate answer sentences (10) – (12) (found via a Google search on 'eat jellyfish'), *jellyfish* corresponds to the ARG2 of *eat*.

(10) Turtles eat jellyfish and they have special hooks in their throats to help them swallow these slimy animals.

(11) Sea turtles, ocean sunfish (Mola mola) and blue rockfish all are able to eat large jellyfish, seemingly without being affected by the nematocysts.

(12) Also, open ocean-dwelling snails called Janthina and even some seabirds have been known to eat jellyfish.

(10) is a valid answer which entails the query since the conjunct can be ignored. (11) also entails the query, since *eat large jellyfish* entails *eat jellyfish* (given that *large* is intersective). Finally, in (12), the context *know* can be taken as truth preserving. Thus the rough match heuristic works correctly for all these cases.

Some more complex cases are illustrated by the example query (13). The candidate answer sentences 14) – (18) were all found via a Google search on 'largest town Cornwall'.

(13) What is the largest town in Cornwall?

(14) St Austell is Cornwall's largest town and a centre of the china clay industry.

(15) In spite of its city statue (sic), Truro is not the largest town in Cornwall; there are several larger agglomerations.

(16) Penzance is the largest town in west Cornwall.

(17) St Austell is the largest town in Cornwall, in terms of population.

(18) Also at [URL], St Austell is said to be the largest town, however the population figure is doubtful. (from Wikipedia Portal talk)

(14) entails the query on the assumption that the predicate corresponding to *in* (_in_p) is treated as compatible with the predicate used to encode the possessive (poss). Possessives, like compounds, correspond to semantically underspecified relations, so this is reasonable, but has to be stipulated.

In (15), the negated context should block the match. As mentioned above, the robust matching approach can be refined by the establishment of general contexts in which the matching is invalidated, including negation and verbs such as *doubt*, *deny* and so on, although this depends on having a list of such verbs. This illustrates a fundamental difference between logical entailment and the robust approach: the robust paradigm requires extra information to block the entailment in a particular context, whereas it is necessary to have extra information about a context to prove an entailment logically. The notion of a negative context can be approximated in (R)MRS via an operator which applies over qeq relationships. As discussed by Copestake et al. (2005), the MRS composition rules guarantee that there is a 'backbone tree' of scopal elementary predications related by qeqs. The same applies to RMRS, although in an RMRS from a shallow analyser, the tree may be less complete. Thus, even in an underspecified RMRS, the scope of negation and sentential complement verbs is available.

(16) demonstrates that whether modifiers can be ignored in the answer depends on the query: while *Penzance is a town in west Cornwall* entails *Penzance is a town in Cornwall*, it is not true that *Penzance is the largest town in west Cornwall* entails *Penzance is the largest town in Cornwall*. Thus the notion of robust matching should really be refined for superlatives.

The last two examples of this set are more subtle, and as far as I know, outside the scope of any current broad-coverage QA system. In (17), the clause *in terms of population* specialises the notion of *large* that is being assumed. The entailment only holds on the assumption that this is also the notion of *large* that pertains in the query. In the final example (18) the context *is said to be* would generally be taken as robustly licensing the entailment, but this is explicitly questioned by the rest of the sentence. Note also that this text is casting doubt on statements expressed in other documents.

To summarise:

- Word overlap takes no account of the context

of the query words in the answer.

- Simple RMRS matching considers the context relating the query words, but no context from the remainder of the sentence. The use of RMRSs constrains the predicate-argument structure, but can result in some matches being missed compared to the bag-of-words approach, unless matching is augmented by an additional predicate hierarchy to allow for cases such as the possessive example.[5]

- Refined RMRS matching checks for specific types of context, such as negation.

- Full sentence-based entailment (as tested by the RTE challenge) takes into account the sentence context, but not document context or inter-document context. For high quality QA results, it may be necessary to look for contradictions elsewhere in the document collection and more generally for indications of trustworthiness of the answer document. It is implausible this sort of context can be obtained by full textual analysis of any sizable document collection using existing language processing techniques.[6]

Thus, in a parallel to the way that RMRS can be viewed as augmenting a lexeme sequence with explicit semantic information, robust matching for QA can be viewed as refining the bag-of-words technique to take advantage of additional semantic markup to come closer to genuine validity.

For applications based on scientific texts, we believe it is more important to take negation into account than is the case for general QA. The chemistry texts that we process on the SciBorg project contain considerable discussion of approaches that did not work. The examples below illustrate this:

(19) Enamines have been used widely as a synthon of α-carbanions of carbonyl compounds, however, the reaction of enamines derived from cyclohexanone and piperidine with [compound1] or [compound2] did not provide the desired products.

(20) Surprisingly, oxidation of diol **24** did not lead to the desired dialdehyde **4**, but to the eight-membered ring lactone **25**.

Detection of hedges is also important: see Medlock and Briscoe (2007). For scientific texts, the relationship between documents is also crucial. We expect to take this into account for IE tasks on SciBorg. For instance, citations may suggest that the current authors are building on some prior work or that they are contrasting their approach with it and this distinction may be very relevant to IE (Teufel, 2005; Teufel et al., 2006).

Note the difference between search/IE/QA and idioms (as discussed in §4.2). The context of the semantic pattern should not affect idiom recognition: for instance, negation does not block the idiomatic interpretation in (21).

(21) I also didn't take much heart from the Gates hearings yesterday.

5 Conclusion

What I have tried to show in this paper is that it is possible to use robust methods with semantic representations that are capable of capturing the information from a 'deep' linguistically-motivated grammar. MRS and RMRS can be used in a variety of approaches and with a large range of applications. I have discussed the operations of matching, merging and munging which we have found useful in a range of contexts, and tried to show that they are of some theoretical interest and not just engineering expedients, but it is clear that there are many more options to explore.

One topic which is important for our future research is the relationship between semantic web languages and (R)MRS. There are many points of potential contact. In some ways, semantic web query answering has more in common with natural language search, QA and IE than it does with database query and most types of formal reasoning. Database query (including via natural language) may rely on a closed world assumption. This allows questions to be answered that involve calculations over a set. For instance, the query in (1), asking for the highest sales in a particular time period, can be answered in a database context by finding all the sales figures for that period and comparing them. This can only be done validly in situations where we can reliably obtain complete information, which is not the case for text-based querying on arbitrary document collections

[5] An alternative is a weighted match where matches for closed-class words and construction relations are given less weight than matches on EPs derived from lexemes.

[6] However, Google PageRank (Brin and Page, 1998) and comparable techniques do provide some indication of reliability and thus act as a rough approximation.

or for many of the semantic web scenarios. Another issue that arises both for natural language query and for the semantic web is the need to allow for different terminology. In semantic web terms, this requires mapping between different schemata: there is an assumption that a meaning preserving mapping must be available for successful query, but finding that mapping may be arbitrarily complex. Even when schemata are officially shared, the conventions of use will tend to vary. Natural language querying, on the other hand, generally does not permit precise mapping between terms, but conversely, for most domains, we know how to express concepts in natural language much better than we do in any formal language. Karen Spärck Jones argued (in the context of her concept of 'Information Layer' (Sparck Jones, 2007)) that connection via words may be good enough for many tasks. The slightly different take on this idea that comes from considering robust broad-coverage semantics is that we can enrich text with extracted formal semantic information (and other annotation) rather than attempting to replace it.

Acknowledgements

Work on this paper was partially funded by EPSRC (EP/C010035/1). I am grateful to colleagues within DELPH-IN and in the Cambridge Natural Language and Information Processing research group. As usual, all errors are my responsibility.

References

Jason Baldridge and G.-J. M. Kruijff. 2002. Coupling CCG and Hybrid Logic dependency semantics. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*. Philadelphia.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics (IWCS-6)*, pages 42–53.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proc. of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP-2005)*.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web*, pages 107–117.

Ulrich Callmeier, Andreas Eisele, Ulrich Schäfer, and Melanie Siegel. 2004. The DeepThought core architecture framework. In *Proceedings of LREC-2004*. Lisbon.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings 7th European Workshop on Natural Language Generation (EWNLG'99)*, pages 86–95. Toulouse.

John Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In R. Dale and K-F. Wong, editors, *Second International Joint Conference on Natural Language Processing (IJCNLP05), Springer Lecture Notes in Artificial Intelligence, Volume 3651*, pages 165–176. Jeju Island, Korea.

Ann Copestake. 2003. Report on the design of RMRS. DeepThought project deliverable.

Ann Copestake. 2007. Semantic composition with (robust) minimal recursion semantics. In *Proc Workshop on Deep Linguistic Processing, ACL 2007*. Prague.

Ann Copestake, Peter Corbett, Peter Murray-Rust, CJ Rupp, Advaith Siddharthan, Simone Teufel, and Ben Waldron. 2006. An architecture for language processing for scientific texts. In *Proceedings of the 4th UK E-Science All Hands Meeting*. Nottingham, UK.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal Recursion Semantics: an introduction. *Research on Language and Computation*, 3(2-3):281–332.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment challenge. In Quiñonero-Candela et al, editor, *MLCW 2005, LNAI Volume 3944*, pages 177–190. Springer-Verlag.

Rebecca Dridan. 2006. *Using Minimal Recursion Semantics in Japanese Question Answering*. Master's thesis, University of Melbourne.

Rebecca Dridan and Francis Bond. 2006. Sentence comparison using Robust Minimal Recursion Semantics and an ontology. In *Proceedings of the Workshop on Linguistic Distances*, pages 35–42. Sydney, Australia.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Dan Flickinger, Jan Tore Lonning, Helge Dyvik, Stephan Oepen, and Francis Bond. 2005. SEM-I rational MT — enriching deep grammars with a semantic interface for scalable machine translation. In *Proceedings of MT Summit X*. Phuket, Thailand.

Anette Frank, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg, and Ulrich Schäfer. 2007. Question answering from structured knowledge resources. *Journal of Applied Logic, Special Issue on Questions and Answers: Theoretical and Applied Perspectives*, 5(1):20–48.

Claire Gardent and Laura Kallmeyer. 2003. Semantic construction in feature-based TAG. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-03)*, pages 123–130. Budapest.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 9–16. Ann Arbor, MI.

Aurelie Herbelot and Ann Copestake. 2006. Acquiring ontological relationships from Wikipedia using RMRS. In *ISWC 2006 Workshop on Web Content Mining with Human Language Technologies*. Athens, Georgia.

Jerry Hobbs. 1985. Ontological promiscuity. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics (ACL-85)*, pages 61–69. Chicago, IL.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *ACL 2007 Workshop on Textual Entailment and Paraphrasing*. Prague.

Ben Medlock and EJ Briscoe. 2007. Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-2007)*. Prague.

Dan I. Moldovan, Christine Clark, Sanda M. Harabagiu, and Daniel Hodges. 2007. Cogex: A semantically and contextually enriched logic prover for question answering. *J. Applied Logic*, 5(1):49–69.

Eric Nichols, Francis Bond, and Daniel Flickinger. 2005. Robust ontology acquisition from machine-readable dictionaries. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. Edinburgh.

Steven Pulman. 1993. The recognition and interpretation of idioms. In *Idioms: Processing, Structure, and Interpretation*, pages 249–270. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Suzanne Riehemann. 2001. *A Constructional Approach to Idioms and Word Formation*. Ph.D. thesis, Stanford University.

Stuart Shieber. 1993. The problem of logical form equivalence. *Computational Linguistics*, 19(1):179–190.

Karen Sparck Jones. 2007. Natural language and the Information Layer. Athena and Lovelace Awards video.

Simone Teufel. 2005. Argumentative zoning for improved citation indexing. In James G. Shanahan, Yan Qu, and Janyce Wiebe (Eds.), editors, *Computing Attitude and Affect in Text: Theory and Applications*, pages 159–170. Springer.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. An annotation scheme for citation function. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*. Sydney, Australia.

Josef van Genabith and Richard Crouch. 1999. Dynamic and underspecified semantics for LFG. In Mary Dalrymple, editor, *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. MIT Press.

Rebecca Watson, Judita Preiss, and EJ Briscoe. 2003. The contribution of domain-independent robust pronominal anaphora resolution to open-domain question-answering. In *International Symposium on Reference Resolution and its Application to Question-Answering and Summarisation*. Venice.

Pinyomi: Dictionary lookup via orthographic associations

Lars Yencken
NICTA Research Lab
University of Melbourne
lljy@csse.unimelb.edu.au

Zhihui Jin and **Kumiko Tanaka-Ishii**
Graduate School of
Information Science and Technology
University of Tokyo
{jin,kumiko}@i.u-tokyo.ac.jp

Abstract

Bilingual dictionaries provide meaning associations between the words of two languages, those of an ideal bilingual speaker. Learners can use these associations to look up foreign equivalents of known native words, but are forced to use script-based lookup methods when faced with unknown foreign words. This paper presents the Pinyomi Chinese-Japanese dictionary interface, which uses a novel method of using associations with the learner's native script to look up foreign language words.

1 Introduction

Bilingual dictionaries can be considered as bipartite graphs, where the nodes are the words of the two languages, and the links between them correspondences in meaning, as shown in Figure 1. Example (a) shows that the general relationship is many-to-many, since words may have many senses, and the distribution of these senses differs from language to language. In this case the French *adresse* shares senses of the English *address* (a description of location; a speech), but can also correspond to *skill* or *dexterity*. This example also highlights that associations of sound and of orthography also occur for a word pair when either features are similar enough.

Where enough similarity occurs, learners can frequently guess part or all of a foreign word's meaning based on its associations to native words. This is a form of beneficial *language transfer* from their native language to their second language. When sound or orthography associations do not coincide with meaning associations, these misleading pairs are called *false friends*. In example (c), the Chinese 手纸 could be naively guessed to mean "letter" by a Japanese speaker, instead of its real meaning "toilet paper".

Regardless of their semantic accuracy, associations can still be useful for lookup, particularly in cases where the lookup target is difficult for the user to input directly, as is the case in Japanese and Chinese[1]. Typically users must painstakingly look up each character in the foreign word by stroke count and radicals before they can look up the word as a whole. Instead, they may query via associated words or characters in their native language, which are easy for them to input.

In this paper we present the Pinyomi system, a dictionary interface for Chinese learners studying Japanese, and Japanese learners studying Chinese, which allows these learners to quickly lookup an unknown word using pronunciation from their own native script. It takes advantage of the script similarities between these languages, allowing users to lookup a foreign word by the orthographic associations that word provides. Since such associations are often noisy, Pinyomi uses a statistical model to recover candidates for the desired foreign word. This form of lookup is significantly more convenient than naive script-based methods.

For example, suppose a Japanese speaker was trying to look up the Chinese word 趣闻. The first character exists in Japanese, and the second is similar to the Japanese 聞. Together these might be read [shumoN] in Japanese. Searching Pinyomi with this reading yields the desired word 趣闻 [qùwén] "interesting news" and its Japanese translation.

[1] In some cases, this can be circumvented on computer by cutting and pasting the unknown word into an electronic dictionary.

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 13–21,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

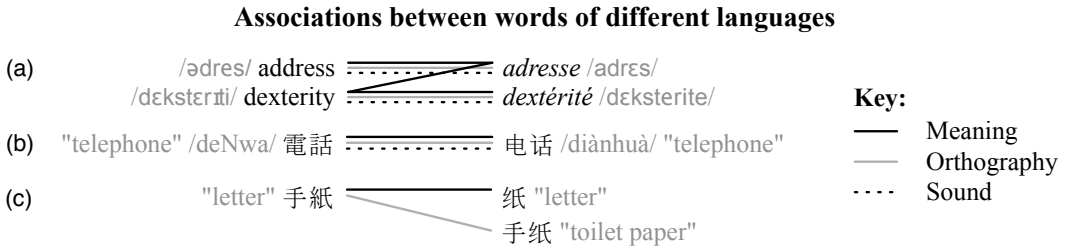


Figure 1: Bilingual dictionaries as bipartite graphs, and different forms of word associations which can occur. Example (a) is English-French, and examples (b) and (c) are Japanese-Chinese.

The remainder of this paper is structured as follows. We firstly introduce related work in transliteration and dictionary lookup (Section 2), before discussing how associations form between the Japanese and Chinese languages (Section 3). We provide an overview of the system and its architecture (Section 4), then delve into the lookup model itself (Section 5). Finally we evaluate our system (Section 6) and discuss its performance (Section 7).

2 Related research

In order to search for a foreign word, Pinyomi requires the user to convert the foreign characters into native characters using any associations they may have upon seeing the foreign characters. The general task of converting a word from one script to another is called *transliteration*. The task of recovering the original word given the converted word is called *back-transliteration*, and is considerably more difficult[2]. It is this latter task which Pinyomi performs to recover a foreign word given a transliteration into the user's native script.

There is much interesting work on these tasks between various language pairs, including English-Japanese (Knight and Graehl, 1998; Brill et al., 2001; Qu et al., 2003), English-Chinese (Li et al., 2004), and many others. The conversion between scripts which Pinyomi users perform has some similarity to transliteration, but is chiefly different its conversion between *logographic* scripts of shared origin.

Since both Japanese and Chinese have a vast wealth of distinct symbols to draw upon in writing, less information is lost in transliterating between these languages. The problem is then closer to Chinese-to-Chinese conversion in difficulty (Halpern and Kerman, 1999). The lack of research in transliterating between this pair also indicates that importing of names and technical terms is less productive compared with other language combinations.

Pinyomi also shares features of high-accessibility dictionaries which allow queries based on noisy or partial user input, a good example being the FOKS dictionary for Japanese (Bilac et al., 2002). FOKS allows a user to guess the compositional reading of a set of characters based on any of their individual readings, and can correct for many common deviations from the correct pronunciation. Pinyomi also complements recent use of associations within monolingual dictionary lookup, aiding writers with the tip-of-the-tongue problem (Ferret and Zock, 2006), by extending this work to more general associations and a bilingual context.

3 Associating Han characters

Pinyomi aims to make use of the natural associations learners form when seeing an unknown word. The type and number of associations that form necessarily depend on the language pair targeted. For Japanese and Chinese, we now consider their writing systems in more detail.

Japanese and Chinese both use a logographic script, called *kanji* in Japanese and *hanzi* for Chinese (where we refer to simplified forms). Kanji in general consist of traditional Chinese characters, with a smaller number of native Japanese additions and simplifications. Kanji and hanzi are thus related historically as branches of an earlier traditional Chinese script. For this reason, we call such characters *Han characters*, and words consisting of such characters *Han words*. For kanji, pronunciation can be expressed in a phonetic *kana* script,

[2]The different phonetic and symbolic inventories of any two languages make transliteration lossy in general. This creates ambiguity for back-transliteration.

and for hanzi the *pinyin* romanisation is typically used (see Figure 2).

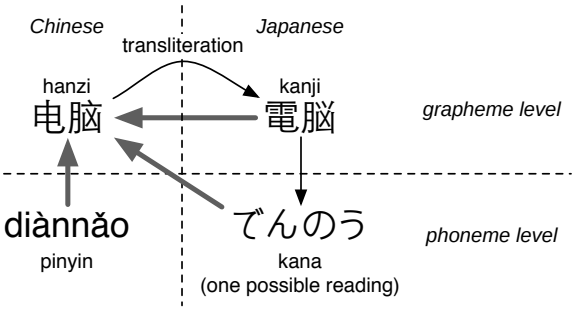


Figure 2: An example of Chinese-Japanese lookup using Pinyomi for the Chinese word meaning "computer". Grey arrows represent lookup methods.

There are two significant types of character-level associations which learners may have when seeing a foreign Han character, shape-based and semantic, which we now discuss with the aid of Table 1. Shape-based associations are made when the shape of a foreign Han character is similar to one from the learner's native language. The simplest example of this is characters shared by both scripts. Other pairs have such similar shapes that they are easily recognisable to natives of one script learning the other. A larger set become associated if some script simplification rules for Chinese are understood. For example, once a learner discovers that 东 is a simplification of 東, it will be very easy for him to associate 冻 with its cognate 凍, and many more follow. These correspond to the "Shape-based" category in Table 1.

Type	Chinese	Japanese
Chinese-only	你 乒	
Shape-based	主 冻	主 凍
Semantic	卖 站	売 駅
Ambiguous	发 馆	発 發 舘 館
Japanese-only		込 峠

Table 1: Examples of Han characters, with and without equivalence.

Pairs similar in shape are usually so because they are *cognates*, modern variants of what was once a single character. Furthermore, since character meanings have been quite stable over time in both languages, the meaning of cognate pairs is usually strongly related. Sub-components of characters also influence whole-character semantics, so a strong similarity of shape may correspond to a similarity of meaning even amongst non-cognates.

On the other hand, there are pairs whose shape may differ significantly, but whose meaning may be very similar. For example, 站 [zhàn] and 駅 [eki] both have the main meaning "station", although 站 can be used for a bus stop, whereas 駅 cannot. This semantic association can take place even though the pair are not cognates[3]. Some cognates may also be difficult to determine due to shape differences. However, given a character in context, the surrounding characters can provide the strong cues needed for such an association to form. These correspond to the "Semantic" category in Table 1.

When potential associations exist with several native characters, the learner may transliterate in several possible ways, forming the "Ambiguous" category in Table 1. This often occurs when more than one native cognate exists, possible in the Chinese-Japanese direction since hanzi have been more simplified than kanji, merging many historically distinct characters together. It also occurs where the meaning in one or both languages has drifted substantially, making room for non-cognate semantic equivalents.

There also exist characters which learners will not associate strongly with any in their native script. This may be particularly true of characters created in the two countries since the time the scripts diverged, such as 你 [nǐ] "you" in Chinese and 込 [ko] "crowded" in Japanese. In Table 1 these cases are marked as "Chinese-only" or "Japanese-only". This effect also occurs when a cognate does exists, but it is much rarer in the other language.

Once the learner has mentally produced such a transliteration, how might they pronounce it? This problem is small in the Japanese-Chinese direction, since most hanzi have a unique pronunciation, but is significant for kanji, which feature many individual readings, and even non-compositional readings in compounds. Figure 3 depicts this ambiguity for individual characters,

[3]站 exists but is rare in Japanese. 駅 is cognate with 驿 in Chinese via the traditional Chinese character 驛 [yì] "remount station".

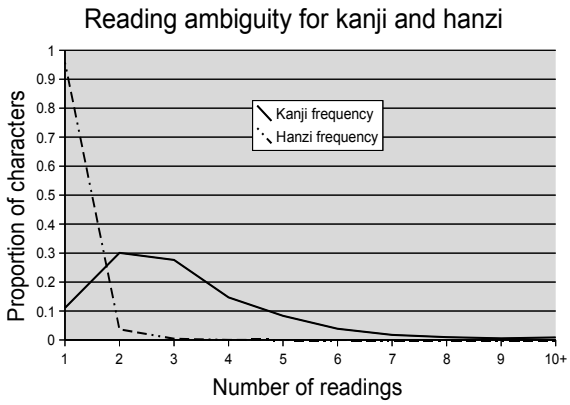


Figure 3: The distribution of reading ambiguity for Chinese and Japanese characters. Hanzi are taken from the GB 2312 set, kanji from JIS X 0208-1990 and JIS X 0212-1990 sets.

where the horizontal axis indicates the number of distinct readings, and the vertical axis indicates the proportion of characters that have this number of readings. For non-word character sequences or rare words, any combination of possible readings could be correct, and should be considered valid inputs.

4 Overall system

In this section, we provide an overview of Pinyomi, firstly from a user's perspective, and secondly from an engineering perspective.

4.1 User interface

Pinyomi presents two user interfaces: a Chinese dictionary supporting lookup by kana, and a Japanese dictionary supporting lookup by pinyin. Figure 4 below indicates the simple layout of results from a Japanese-Chinese query, with results scored by probability.

In the reverse direction, to look up the Chinese 电话, a Japanese person can follow shape similarity or simplification rules to determine correspondence with the Japanese 電話. They thus enter its reading, "でんわ" [deNwa] into the input box, and click "Search". The results displayed contain the desired Chinese word and its translation.

Despite a Japanese learner's natural ability to transliterate from Chinese to Japanese, there will be some characters for which no association is strong. In these cases, the user replaces that character's reading with a special wild-card character

Figure 4: Pinyomi's Japanese-Chinese dictionary interface, where the user is searching for 電化.

"?", and can still find the target word.

4.2 Architecture

Details of candidate generation and scoring are provided in the following section, but an overview of Pinyomi's construction is as follows. For each foreign Han word in the dictionary, we:

1. Determine the word's corpus frequency.
2. Exhaustively determine and store transliteration candidates for the word.
3. For each transliteration candidate, exhaustively determine and store potential readings for this candidate.

This procedure constructs the database, which is queried at run-time by the web interface. The resources used for the previous steps are:

- A Chinese-Japanese and Japanese-Chinese bilingual dictionary.
- A Japanese corpus: we used 220 MB from the Nikkei corpus, taken from the Nikkei financial newspaper.
- A Chinese corpus: we used 200 MB from the Peking University corpus for Chinese, consisting of a wide selection of popular print media (Center for Chinese Linguistics, 2007).
- A reading dictionary for Japanese: the Kanjidic dictionary[4].

[4] `http://www.csse.monash.edu.au/~jwb/kanjidic.html`

- A GBK pinyin reading table for Chinese.
- A table of hanzi and kanji equivalence candidates (Goh et al., 2005).

We now describe the scoring model in detail.

5 Candidate generation and scoring

5.1 General model

The general model is provided in Equation 1 below, where s is the word a user would like to look up, t is its transliteration into their native language, and r is the reading of that transliteration in their native phoneme representation. The model determines which candidates s to display to the user if they enter r as their query, and orders them by $\Pr(s|r)$.

$$\begin{array}{rcl} s & : & \text{word in foreign language} \\ \phi(s) & : & \text{all transliterations candidates of s} \\ t & : & \text{a transliteration candidate} \\ r & : & \text{reading in native language} \end{array}$$

$$\begin{aligned} \Pr(s|r) &\propto \Pr(r|s)\Pr(s) \\ &= \Pr(s)\sum_{t\in\phi(s)}\Pr(r,t|s) \\ &= \Pr(s)\sum_{t\in\phi(s)}\Pr(r|t,s)\Pr(t|s) \\ &= \Pr(s)\sum_{t\in\phi(s)}\Pr(r|t)\Pr(t|s) \qquad (1) \end{aligned}$$

In the first line, $\Pr(r)$ is not important for ordering results, so it is omitted. After the user transliterates the foreign word s to native pseudo-word t, the reading r chosen by the user is only dependent on t, so we can simplify $Pr(r|t,s) = Pr(r|t)$.

The final line in Equation 1 requires three submodels to be defined, namely:

- A language model to calculate $\Pr(s)$.
- A transliteration model to calculate $\Pr(t|s)$.
- A reading model to calculate $\Pr(r|t)$.

We now discuss these additional models.

5.2 Language model – $\Pr(s)$

$Pr(s)$ is the probability that the user will look this foreign word up, relative to other words in the dictionary. This may not coincide fully with corpus frequency, since some common words are unlikely to be looked up if their meaning is obvious to these learners. Nonetheless, corpus frequency is a good approximation which makes no assumptions about the user's proficiency level.

5.3 Transliteration model – $\Pr(t|s)$

We aim to model how users will perform their mental transliteration, using the associations mentioned in Section 3. In fact there are two related problems this model must solve: what candidates to use, and how to score them.

Ideally, we would use a parallel corpus between Chinese and Japanese to automatically extract transliterations, and then use this data to solve both of these problems simultaneously. By assuming the process is Markovian, we could train an n-gram model to perform context-sensitive transliteration. However, in the absence of such a corpus, we are limited to simpler approximations.

To determine the candidate set, we use a mapping table provided by Goh et al. (2005), which was constructed by mostly using cognate pairs, but also shape-similarity, and semantic correspondence – the same associations we discussed earlier. This mapping table itself is incomplete: it is one-to-many from hanzi to kanji, although general associations should be many-to-many. This makes the model ambiguous when used to transliterate from hanzi to kanji, but not in the reverse direction. Nonetheless, it is useful as a starting point for future work.

Given foreign Han word $s = s_1 \ldots s_n$, we model the user's transliteration into the native pseudo-word $t = t_1 \ldots t_n$ on a character by character basis.

$$\begin{aligned} \Pr(t|s) &= \Pr(t_1 \ldots t_n | s_1 \ldots s_n) \\ &= \prod_{i=1}^{n}\Pr(t_i|s_1 \ldots s_n) \\ &= \prod_{i=1}^{n}\Pr(t_i|s_i), \quad t_i \in \phi(s_i) \qquad (2) \end{aligned}$$

For each foreign Han character, there may be several associated native characters which they might use. Here we assume that the t_i is only dependent on s_i, so we reduce $\Pr(t_i|s_1 \ldots s_n)$ to $\Pr(t_i|s_i)$. For each foreign Han character s_i, the set of corresponding native Han character candidates $\phi(s_i)$ can be found in the mapping table. The mapping probability is estimated by using relative corpus frequency:

$$\Pr(t_i|s_i) = \frac{\#t_i}{\sum_{t_j \in \phi(s_i)} \#t_j}$$

Since the mapping table only contains one-to-many mapping from hanzi to kanji, when we convert a kanji s_i to hanzi t_i, we only have one choice, so in this case $\Pr(t_i|s_i) = 1$.

When a foreign characters s_i yields no associations, the user can instead use a wild-card character "?". Thus any potentially unreadable character should have "?" as a transliteration candidate. This occurs when: a) all potential associations are with rare, unknown characters, or b) the shape is very different, making the association difficult to form. To estimate the native characters that each user would know, we used high-school-level character sets for each country, as determined by government education standards.

Suppose that k characters in total can be potentially replaced by "?". Then when the user maps a foreign character s_i to "?", we simply calculate the mapping probability using a uniform distribution $\Pr(?|s_i) = \frac{1}{k}$.

5.4 Reading model – $\Pr(r|t)$

Given the pseudo-word t in native language, the user must choose a reading for this word, as discussed in Section 3. Here we use a simple unigram model, which suffices to distinguish candidates in both lookup directions. We leave the possible impact of higher order models as an open question for future research.

$$\begin{aligned} \psi(t) &: \text{all readings of t} \\ \Pr(r|t) &= \Pr(r_1 \dots r_n | t_1 \dots t_n) \\ &= \prod_{i=1}^{n} \Pr(r_i | t_1 \dots t_n) \\ &= \prod_{i=1}^{n} \Pr(r_i | t_i), \quad r_i \in \psi(t_i) \end{aligned} \tag{3}$$

A crucial feature of the Pinyomi kana interface is that it requires kana readings to be input by the user in character-by-character, space-separated manner. This rules out the use of compositional readings, even where the t sequence forms a natural word in Japanese. Adding such a constraint on the user is acceptable, since the users are native speakers, and are highly likely to know many readings for a given kanji.

	No wild-cards		Using ?		
$\|w\|$	Best	Random	Best	Random	%
1	9.41	12.40	56.31	58.50	25.7
2	1.63	2.02	4.48	6.04	49.3
3	1.01	1.01	1.13	1.19	12.2
≥4	1.00	1.00	1.00	1.01	12.7
all	3.47	4.43	16.94	18.28	100

Table 2: Mean rank by word length, using either the most likely kana query, or a random kana query for each Chinese word in the dictionary.

Japanese is non-segmenting, and each kanji can have a readings of various length. For this reason, determining corpus frequency for our unigram model was slightly more complicated than in the pinyin direction. We took frequency counts for kanji readings from the 1990 Nikkei corpus, which was segmented and annotated with readings by Chasen, and finally grapheme-phoneme aligned using a simple TF-IDF based alignment model (Yencken and Baldwin, 2005). Counts for compositional readings were discarded, readings outside those provided by Kanjidic were filtered as noise or obscure name readings, and any remaining readings had their frequency counted. These frequencies were used to train the unigram model above.

6 Evaluation and Analysis

We evaluate Pinyomi in two primary ways. The first is some basic analysis of the reading database itself, and the second is through a user experiment, where we asked users to give native readings for foreign Han words.

6.1 Reading set analysis

For each word in the Chinese dictionary, we selected from our model of Japanese readings the most probable reading, and queried the database to determine the rank of the original word. We then repeated, but with a random reading from our model. Averaging this rank over all words in the dictionary gives the results in Table 2. In this table, the "No wild-cards" category used readings without "?", whereas the "Using ?" category replaced a query reading with "?" whenever possible. This analysis was only performed for Chinese lookup by kana, since this direction is more ambiguous and thus more difficult.

Notably, the mean rank is quite large for single

character words, of which there are many more in Chinese than Japanese. However, in both cases the mean rank was quite small for words of length 2 or greater, as queries became more constraining. The "Best" reading was intended to capture an ideal case, and the "Random" reading a normal case, however the difference in ranking between the two is small. This indicates that transliterating appropriately is more important than choosing a good reading candidate, since any reading candidate is sufficiently constraining to give a decent ranking.

The very large ranking given to one-character words looked up by wild-card indicate that using "?" alone is simply not constraining enough. Without using wild-cards, the ranking is still not ideal, due to the large number of homophone kanji in Japanese. This lookup method thus is no replacement for looking up individual characters, but instead has large advantages for words which are compounds. For lookup of Japanese words, the frequency of kanji compounds which are one-character is instead at 3%, reducing the problem.

6.2 Reading experiment

Although simulated queries give some information, we are interested in the real-world performance of the system. For this reason, we conducted two symmetric user experiments where we firstly explained the system, then asked users to provide search queries in kana (for Japanese participants) and pinyin (for Chinese participants) for foreign Han words.

For stimulus, 100 random Han words of length two or greater were taken from each of the two bilingual dictionaries, of which 30 were randomly presented to each user. Each user entered between one and three readings in their native language, and told to use "?" if they couldn't read a particular foreign character. Reading input boxes were segmented so that users gave readings on a per-character basis.

In total, 28 Chinese native speakers and 12 Japanese native speakers completed the experiment, most of whom had never studied each-other's language before. These users covered 98 stimulus for kana, and all 100 for pinyin[5].

To determine how well Pinyomi performs on these queries, we performed each query on the Pinyomi system. Each word successfully found has a rank, and by averaging the rank over these words we get a mean rank statistic. We calculated this statistic using just the first reading the user gave, and also using all three. In the latter case, we only used an additional reading when lookup failed, simulating repeated attempts to find the same word. The percentage of queries where the query located the desired word is indicated by the coverage, as given in Table 3.

Method	r	c	n	r/n
Kana	1.99	0.64	10.01	0.20
Pinyin	1.30	0.83	2.54	0.51

Table 3: Rank and coverage for the readings given by experimenters. r is the mean rank of a successful queries, c is the coverage, and n is the mean number of candidates returned.

The mean rank statistic for both lookup methods is very good, indicating that when the query is successful, the desired word is likely to be in the top few results. To evaluate the ranking independently of the number of candidates returned, we consider the relative ranking r/n. For kana lookup, the desired word is typically ranked well in the top 20% of results, compared to pinyin lookup, where the word is in the top 51% of results. The low number of pinyin candidates prevents this from being problematic, but if future enhancements increase the number of candidates, care may be required to maintain a high mean rank.

Of primary concern is the low coverage for kana queries. In order to improve coverage, we need to determine what the cause of the failed queries is.

6.3 Error Analysis

In order to better understand the gaps in coverage, we asked a native Japanese speaker and a native Chinese speaker to analyse the failed kana and pinyin searches respectively, grouping into common error types. The results of these analyses are provided in Table 4.

The following error categories are used. "Reading ambiguity" refers to typical alternations used when native speakers articulate the pronunciation of a word or psuedo-word. Chinese speakers have trouble distinguishing between two close pinyin sound representations, for example zhàn and zàn. Japanese has reading alternation effects which occur when characters are combined in a compound.

[5]Due to the method of random allocation of stimulus to each user, coverage of all stimulus words was not guaranteed. Two words in the kana direction were not covered by any user responses for this reason.

Error Type	Pinyin	Kana
Reading ambiguity	37.6%	14.5%
Radical or shape	22.0%	43.5%
Unknown character	10.8%	25.8%
Miscellaneous	29.6%	16.1%
Total	100.0%	100.0%

Table 4: Error analysis for failed pinyin and kana searches.

For the Chinese word 森林, a Japanese user entered "もりばやし" [mori bayashi] instead of "もりはやし" [mori hayashi]. This effect is called *sequential voicing*, and is one of several in this error category.

"Radical or shape" indicates an error in our transliteration model, since the user formed an unexpected character association not present in our table. "Unknown character" refers to the use of "?" for a character which Pinyomi mistakenly predicted the user could read. For example, one subject had the query "? よう" for the foreign word 称雄, even though the first character 称 is reasonably frequent in Japanese. All other errors are "Miscellaneous", and include typing mistakes and various unclear cases considered as noise.

7 Discussion and future work

A potential criticism of this work is that whilst it provides better accessibility for languages pairs that are quite close, where often learners could accurately guess a foreign word's meaning, perhaps our efforts could be better spent on more distinct language pairs. However, even with languages that share words of identical written form, their senses, frequency of use, pronunciation and connotations may still differ wildly. Further, there is indication that guessing *then* consulting a dictionary has advantages in both retention and accuracy over guessing alone (Fraser, 1999). The dictionary is still a time-consuming staple for these learners.

Given script similarity, why is the coverage firstly asymmetrical and secondly not higher? Since Japanese and traditional Chinese share a vast number of characters, and speakers of simplified Chinese also know some traditional characters, they have a strong advantage over Japanese learners of Chinese. Hence their coverage is 83%, compared to 64% for the latter.

As to the general coverage level, we note that the experiment participants overwhelmingly had no previous exposure to the foreign language. In this light coverage is remarkably high. Furthermore, coverage should improve as learners pick up more cognate relationships, although more extensive evaluation is needed to explore the extent of this improvement. For non-learners, our evaluation suggests a theoretical ceiling coverage of around 95% in both directions[6].

Reading error correction is not difficult, and has precedent in the FOKS dictionary system (Bilac et al., 2002). The main challenge lies in reducing "Radical or shape" errors, particularly important for improving Chinese-Japanese lookup. For Pinyomi, use of sub-components seems the most promising avenue to explore.

Finally, this general approach of traversing associations could be extended to other sufficiently similar language pairs. Pairs which share many cognates are ideal, because of the fertile phonetic, orthographic and semantic similarity neighbourhoods they provide. It remains an interesting question as to how to best leverage these similarity associations for each language pair in order to achieve better outcomes for learners or cross-lingual applications.

8 Conclusion

The Pinyomi dictionary system provides a useful and innovative method for dictionary lookup, using the natural associations that learners build as they learn a second language. In this way it achieves excellent rank and decent coverage, even for non-learners, and provides convenient and accessible lookup of Japanese and Chinese words.

Acknowledgements

We are very grateful to Mrs Chooi-Ling Goh for her kindness in providing us with the kanji/hanzi conversion table from Goh et al. (2005), which greatly aided the construction of this system.

References

Slaven Bilac, Timothy Baldwin, and Hozumi Tanaka. 2002. Bringing the dictionary to the user: the FOKS system. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 85–91. Taipei, Taiwan.

Eric Brill, Gary Kacmarcik, and Chris Brockett. 2001. Automatically harvesting Katakana-English term pairs from search engine query logs. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, pages 393–399. Tokyo, Japan.

[6] Assuming that all correctable errors are modelled and compensated for.

Center for Chinese Linguistics. 2007. Chinese corpus. Visited 2007, searchable from `http://ccl.pku.edu.cn/YuLiao_Contents.Asp`.

Olivier Ferret and Michael Zock. 2006. Enhancing electronic dictionaries with an index based on associations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 281–288. Sydney, Australia.

Carol A. Fraser. 1999. Lexical processing strategy use and vocabulary learning through reading. *Studies in Second Language Acquisition*, 21(02):225–241.

Chooi-Ling Goh, Masayuki Asahara, and Yuji Matsumoto. 2005. Building a Japanese-Chinese dictionary using kanji/hanzi conversion. In *Proceedings of the 2nd Internatioal Joint Conference on Natural Language Processing*, pages 670–681. Jeju Island, Korea.

Jack Halpern and Jouni Kerman. 1999. The pitfalls and complexities of Chinese to Chinese conversion. In *Proceedings of the 14th International Unicode Conference*. Cambridge, Massachusetts, USA.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 159–166. Barcelona, Spain.

Yan Qu, Gregory Grefenstette, and David A. Evans. 2003. Automatic transliteration for Japanese-to-English transliteration. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 353–360. Toronto, Cananda.

Lars Yencken and Timothy Baldwin. 2005. Efficient grapheme-phoneme alignment for Japanese. In *Proceedings of the Australasian Language Technology Workshop 2005*, pages 143–151. Sydney, Australia.

Graphical Query for Linguistic Treebanks

Steven Bird
Department of Computer Science
University of Melbourne
Victoria 3010, Australia
sb@csse.unimelb.edu.au

Haejoong Lee
Linguistic Data Consortium
University of Pennsylvania
Philadelphia PA 19104-2653, USA
haejoong@ldc.upenn.edu

Abstract

Databases of hierarchically annotated text occupy a central place in linguistic research and language technology development. We describe a new approach to tree query which we call "Query by Annotation". Users express a query by annotating a tree, and the annotation is compiled into an expression in a path language. The result trees are overlaid with the original query, permitting the user to see why they match. Since queries and results are annotated trees, users can easily refine and resubmit their queries. The approach to Query by Annotation is motivated and exemplified using databases of linguistic trees, or treebanks.

1 Introduction

Large repositories of text and speech data are routinely collected, curated, annotated, and analyzed as part of the task of developing and evaluating language technologies. These repositories contain millions of words of text, along with various annotations at the levels of phonetics, prosody, orthography, syntax, dialog, and gesture. The annotations are often hierarchical in nature, and are anchored to extents of text or speech. The hierarchical annotations can be stored as ordered trees.

Empirical investigations of hierarchically annotated linguistic data typically involve the identification or extraction of substructures, according to their position within the overall structure and their internal organisation. Consider the following kinds of access to syntactic trees: find instances of the dative construction (a verb phrase containing a verb followed by two noun phrases); extract all simplex noun phrases (noun phrases that do not contain other noun phrases) collect prepositional phrase attachment data (verb, preposition, head noun, attachment node). In each of these cases, and many others we could list, we may want to find instances of theoretical interest, or create a derived corpus, or extract features for training an automatic classifier. And in each case, we would like to be shielded from the physical storage of the corpus as a directory tree of formatted text files. All these needs are served by linguistic query languages.

Over a dozen linguistic query languages have been proposed, each with its own specialised interpreter for evaluating queries against a corpus (Rohde, 2001; König and Lezius, 2001; Kepser, 2003; Resnik and Elkiss, 2003; Mírovský, 2006; Lai and Bird, 2004). For concreteness, the work presented here will be based on the LPath language. This language has full first-order expressiveness (Lai, 2005), and can be translated into SQL for efficient evaluation (Bird et al., 2006). Although we have selected LPath, our approach is independent of the underlying tree query language and tool infrastructure. Also for concreteness, our examples will be drawn from English syntax and the Penn Treebank (Marcus et al., 1993). However, our approach is independent of the linguistic domain and data source, and can be applied to any hierarchically-annotated time-series data.

This paper presents a new approach to querying linguistic trees, namely *Query by Annotation* (QBA). In this approach, a query is expressed as an annotation of a given tree. Such a query denotes a set of trees which are similar to the given tree in precise ways. It is related to an existing approach to database query known as Query by Example (Zloof, 1977). It differs from XQBE, XQuery by Example (Braga et al., 2005), in that it only covers the selection component, and it is tailored for the specific domain of linguistic tree query. It differs from graphical interfaces to XPath that permit users to type an XPath query and see node-sets highlighted on an instance document, in

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 22–30,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

that it supports direct annotation of a query on a tree, displayed in the customary form of a parse tree.

QBA provides users with several benefits relative to direct use of a path language. First, QBA provides a high-level interface to a path query language, avoiding the need for users to learn a query syntax. Second, QBA queries are not created *ex nihilo*, but by annotating an object from the domain. Users find it easy to express queries of the form: "find me more trees that are like this one in the specified way." Third, result trees can be automatically overlaid with the original query, which means that queries and results are of the same type, namely annotated trees. A user sees why the query matched, and can edit the annotation to refine the query.

The main contributions of this work are as follows: a new approach to graphical, semi-structured query is presented, in which examples are annotated with a query graph, and queries are translated into SQL, and results are annotated with the original query; an application to linguistic databases is described, motivating and exemplifying the approach; and an implementation is reported, involving translation steps from an annotated tree to a path language, thence to SQL for evaluation.

This paper is organised as follows. In Section 2 we review key background topics including linguistic annotation, corpus curation, and query by example. In Section 3 we present our approach to query by annotation, and in Section 4 we show how annotated queries are translated into the LPath language, then in Section 5 we explain how queries are overlaid on result trees. Section 6 describes a prototype implementation, and Section 7 reports our conclusions.

2 Background

2.1 Linguistic Annotation

Linguistic databases consist of time-series data together with structured annotations. The time-series data represents an external linguistic artefact, and takes the form of a text or recording. The relationship between the primary data and its annotations is shown schematically in Figure 1.

A common data model for linguistic annotations is a labelled, ordered tree. (The nodes of the tree are ordered, by virtue of the linear ordering of the time-series data.) A natural candidate for representing trees is XML, as shown in Figure 2(a). These structures can be stored efficiently in relational form, using a span-based representation as shown in Figure 2, following (Bird and Liberman, 2001; Bird et al., 2006).

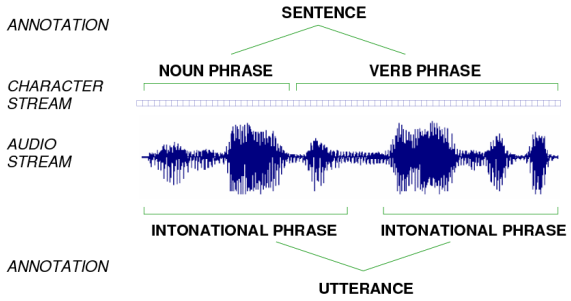


Figure 1: Linguistic Annotation: Structured Coding of Extents of Time-Series Data (e.g. Character Data, Audio Data)

2.2 Curating Treebanks

Treebanks are typically created over an extended period. Initial processing is done using a statistical parser, followed by substantial manual editing. During the course of this activity, highly-specific annotation conventions are developed; these conventions are further elaborated as new constructions are encountered. Thus, treebank creation involves significant *curation* work.

A typical part of the curation workflow is for a linguistically trained annotator to discover an incorrect parse (generated by a statistical parser), and then try to identify all other occurrences of the faulty structure and to rectify them as necessary. In the earliest days of treebank development, such curation was done by editing labelled bracketings in a text file, aided with editing macros and Perl scripts. Not surprisingly, this approach did not scale well. A common response has been to develop a tree query language.

2.3 Querying Linguistic Trees

Many tree query languages have been developed for parsed corpora, e.g. tgrep, TIGERSearch, fsq, LSE, Netgraph (Rohde, 2001; König and Lezius, 2001; Kepser, 2003; Resnik and Elkiss, 2003; Mírovský, 2006). For a survey, please see (Lai and Bird, 2004). Current graphical tree query interfaces permit users to draw partial trees from

```
<S>
  <NP lex="I"/>
  <VP>
    <V lex="saw"/>
    <NP>
      <NP>
        <Det lex="the"/>
        <Adj lex="old"/>
        <N lex="man"/>
      </NP>
      <PP>
        <Prep lex="with"/>
        <NP>
          <Det lex="a"/>
          <N lex="dog"/>
        </NP>
      </PP>
    </NP>
  </VP>
  <N lex="today"/>
</S>
```

(a) XML Representation

left	right	depth	id	pid	name	value
1	10	1	2	1	S	
1	2	2	3	2	NP	
1	2	2	3	2	@lex	I
2	9	2	4	2	VP	
2	3	3	5	4	V	
2	3	3	5	4	@lex	saw
3	9	3	6	4	NP	
3	6	4	7	6	NP	
3	4	5	8	7	Det	
3	4	5	8	7	@lex	the

...

(b) Relational Representation *T*

Figure 2: Linguistic Tree Representations

scratch, and the fragments are matched against trees in the treebank (Resnik and Elkiss, 2003; Mírovský, 2006). These existing approaches have shortcomings in the areas of interactiveness and expressiveness.

First, tree queries are seldom single-shot, but must be successively refined. As with web queries – where the original query is displayed in editable form along with the result – formulating suitable linguistic tree queries would be aided by a user interface that supports interactive query refinement.

Second, tree queries generally involve a variety of transitive relations (e.g. precedence) and negated expressions which cannot be expressed by drawing tree fragments.

2.4 Query by Example

Query by Example (QBE) was an early approach to user-friendly database query that shielded the user from the SQL query language (Zloof, 1977). Users search for data by partly completing a form, and this is then interpreted as an SQL query and submitted to the database engine. In short, the user initiates a search simply by providing an example of what they are seeking.

QBE is a natural way to explore and curate treebanks, given the typical workflow of progressing from an instance to a set of "similar" instances. Of course, the notion of tree similarity changes from one case to the next, and may depend on a combination of factors including structure, categories, and terminals. Thus we should not use a whole tree as the basis for identifying similar trees. Instead, we propose to *annotate* a tree in order to specify which properties must hold for any "similar" tree.

2.5 LPath

LPath is a language for querying linguistic trees which extends XPath (Clark and DeRose, 1999) with new primitive horizontal tree navigation axes, subtree scoping and edge alignment, summarised in Table 1. This language has full first-order expressiveness (Lai, 2005), and can be compiled into SQL for efficient evaluation (Bird et al., 2006). Here is the translation of the query `//A//B` into SQL:

```
select T1.* from T T0, T T1
  where T0.type='syn' and T0.name='A'
  and T1.type='syn' and T1.name='B'
  and T0.sid=T1.sid and T0.tid=T1.tid
  and T0.l<=T1.l and T0.r>=T1.r and T0.d<T1.d
```

Many useful queries turn out to be simple to express in this path language, owing to the fact that linguists identify tree nodes and relationships between nodes by reference to structurally local information.

3 Query by Annotation

Query by Annotation is an approach to tree query in which a user annotates an existing tree – the "base tree" with a query. The base tree may be any tree found in the treebank by browsing or

Table 1: LPath Navigation Axes

Type	LPath Axis	Abbreviation	Closure	Core XPath Support
Vertical	child	`/`		$\surd$
	descendant	`/descendant::`	$/^+$	$\surd$
	parent	`\`		$\surd$
	ancestor	`/ancestor::`	$\backslash^+$	$\surd$
Horizontal	immediate-following	`-⟩`		$\times$
	following	`--⟩`	`-⟩`$^+$	$\surd$
	immediate-preceding	`⟨-`		$\times$
	preceding	`⟨--`	`⟨-`$^+$	$\surd$
Sibling	immediate-following-sibling	`=⟩`		$\times$
	following-sibling	`==⟩`	`=⟩`$^+$	$\surd$
	immediate-preceding-sibling	`⟨=`		$\times$
	preceding-sibling	`⟨==`	`⟨=`$^+$	$\surd$
Other	self	`.`		$\surd$
	attribute	`@`		$\surd$

by an earlier search. The query is an annotation of the base tree in which a subset of the nodes are selected. Lines are drawn between pairs of selected nodes to indicate structural relationships such as "descendent" and "following" that should hold true in any results from a new query. Node labels and attributes are modified as necessary. The result of a query by annotation is a collection of trees, each annotated with the original query. This section describes the graphical elements of the query interface and shows how they correspond to LPath components.

3.1 Axes

The most basic component of a query is a relation between a pair of tree nodes. The inventory of atomic queries is shown in Figure 3, along with translations into LPath.

Observe that the expected relation can be inferred from the base tree. In the context of a graphical interface this saves effort because the user can simply connect nodes without needing to specify the relations. Users can override this default interpretation by clicking on the line to cycle through its possible interpretations, as shown in Figure 4.

3.2 Filters, nodes and attributes

We have observed that users often pose a query which generates far too many results. Evidently the user is not aware of the variety of data contained in a treebank. There are four main ways a result set can be narrowed. First, the user can make node relations more specific (e.g. change "descendent" to "child"). Second, the user can specify node attributes (e.g. mark an NP node as temporal by adding the `TMP` attribute). Third, the user can edit the existing query, adding new edges to more narrowly describe the desired result set. Finally, the user can add new negated edges to remove trees – such as the one currently being viewed – from the result set.

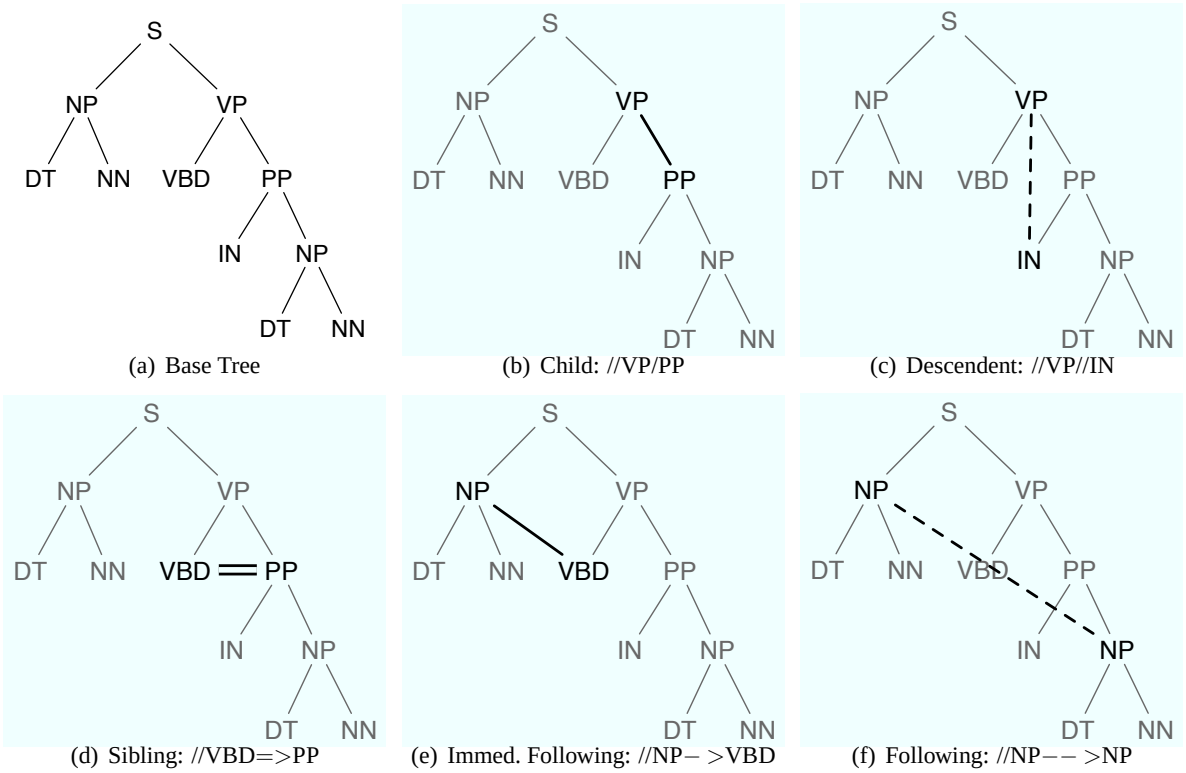
3.3 LPath alignments and scopes

The LPath language has additional features that make it more expressive than XPath. Two of these are alignment and scope, linguistically important properties that need to be represented in the graphical query.

LPath permits queries that stipulate the left- or right-alignment of a subtree within the scope of some ancestor node (e.g. to find a prepositional phrase that is final within an ancestor verb phrase). The GUI makes this expressiveness available by permitting users to right-click on a node and toggle the alignment information.

LPath also permits queries to specify that subexpressions remain within the scope of a particular node. All downward navigations – the "child" and "descendent" axes – introduce a new scope. Subsequent horizontal navigations remain inside the scope of the dominating node iff the corresonding node in the base tree also falls under that node.

For example, in the tree in Figure 3(a), a query starting at the left `NP`, which goes down to the child `DT` then across to the following `NN` has the following scope, by default: `//NP{/DT=⟩NN}`. A query



(a) Base Tree (b) Child: //VP/PP (c) Descendent: //VP//IN

(d) Sibling: //VBD=>PP (e) Immed. Following: //NP− >VBD (f) Following: //NP−− >NP

Figure 3: Translation of QBA Primitives to LPath Axes

which starts from the same NP, then goes down to the child DT as before, then across to the following VBD actually leaves the scope of the NP, and the scope is as follows: //NP{/DT}--⟩VBD.

In the graphical interface, the depth of scope nesting is indicated using a superscript integer. The user can toggle its value to expand or shrink the scope, thereby constraining or relaxing the query (respectively).

4 Query Translation

4.1 Approaches to translation

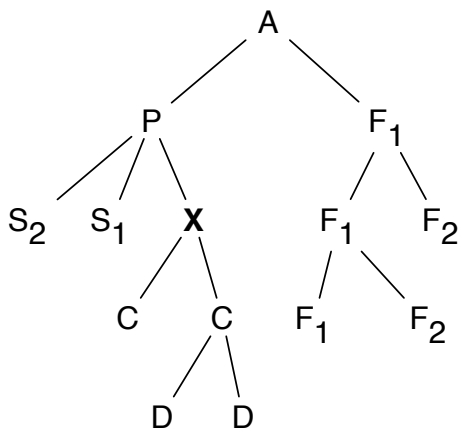
Several approaches to query translation have been investigated. Perhaps the most obvious is direct graph matching, in which a query is matched against a tree using powerful graph matching techniques (Messmer and Bunke, 1998). However, this approach is inadequate, for two simple reasons. First, the query graph is not a subgraph of the result tree. The transitive axes such as "descendent" are not explicit in the treebank. Second, queries involve negation and it is meaningless to match negated edges against actual edges in a treebank. In these respects, queries are not partial trees, but partial *descriptions* of trees.

Another approach is to use first-order logic over trees or annotation graphs (Bird and Liberman, 2001). In the case of trees, direct translation is not possible as the "immediate following" axis cannot be expressed (it would involve an arbitrary number of joins, inexpressible in a first order language). In the case of annotation graphs the dominance axes cannot be expressed (these also involve an arbitrary number of joins to navigate from an edge to another edge included within its span).

Instead, we convert the graphical queries to LPath, and to use the existing LPath query interpreter for onward translation from LPath to SQL (Bird et al., 2006). This approach imposes the restriction that the query graph must be connected, and we have not found this restriction to pose any problems in practice. (The restriction is conveniently implemented in the user interface, as it implements the notion of an active node, and new edges can only be added by linking back to this active node.)

4.2 Linear queries

So far we have seen how atomic queries are translated. The interpretation of individual edges has a well-defined default, and various alternatives that are selected via the user interface (Figure 4). Con-



target	default axis	other axes
A	\\	\
P	\	\\
S_1	<=	<==, <-, <--
S_2	<==	<=, <-, <--
F_1	->	-->, =>, ==>
F_2	-->	->, =>, ==>
C	/	//
D	//	/

Figure 4: Default and Alternative Interpretations of QBA Primitives Relative to Node X

sequently, we will abstract away from the identity of each axis and focus on the structure of complex queries.

The next step in increasing query complexity is a query involving two relations r_1 and r_2 and a single shared node (Figure 5(a)). The translation can start at either end of this path, and simply generate a query of the form: `Ar1Br2C`. This method generalises to linear queries of arbitrary length. It is immaterial which end we start from, as the result of a query is always a whole tree, not a set of nodes (as is the case for XPath queries).

4.3 Branching queries

In general, the structure of a query graph is a *free tree*, a tree with no specified root node and no sibling order. The smallest possible branching query involves three relations r_1, r_2 and r_3 with a single shared node, in a Y structure. The interpreter breaks this structure into a linear component and a branch (Figure 5(b)). The linear component can be translated as before. The branch is also linear and can be translated in the same way. The final step is to connect the pieces together. This is done using a *filter expression*, an expression contained inside brackets and anchored at a particular node: `Ar1B[r3D]r2C`. In this expression, `B` is located at the centre, and is related to `A`, `D`, and `C`.

Negated branches are also handled in this way. Thus if a negated edge links `B` and `D` we would have: `Ar1B[notr3D]r2C` (Figure 5(c)). Multiple branches emanating from a single node can be expressed using conjunction within the filter expression. We use nested filter expressions to translate branches upon branches (Figure 5(d)). Now, given that the linear components of queries can have arbitrary length, and branch points can occur at any node, we can translate query graphs of arbitrary complexity.

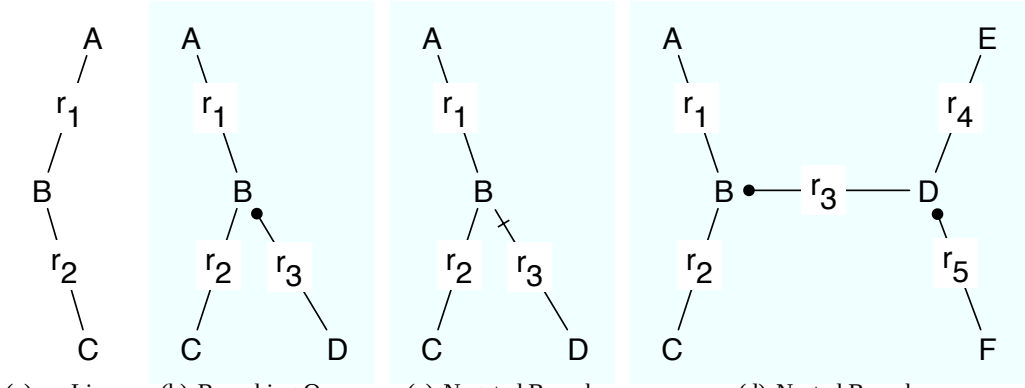
5 Result Overlay

In order to help the user identify where the query matches with the result tree, the graphical query that the user has drawn is overlayed over the new tree. This also helps the user to refine the query, which otherwise has to be redrawn from scratch.

5.1 Database query and result rendering

The graphical query drawn by the user is translated into an LPath query. This in turn is translated into an SQL query by an LPath-to-SQL translator (Bird et al., 2006) (source code available from `http://nltk.org/nltk_contrib/lpath/`). The obtained SQL query is sent to a remote database server and a result table is sent back. The result table contains a set of rows. Each row is a node in a tree that matches with the query, and it also corresponds to the last node of the original LPath query.

Unique tree ids are retrieved from the result table. For each of the ids, as requested by the user, the entire tree is retrieved (i.e. a set of rows, each for a node from the tree in question). This table is transformed to a tree and rendered on the display. Node ids in the original table are kept during query translation to permit overlay of the original query in the graphical display.



(a) Linear Query (b) Branching Query (c) Negated Branch (d) Nested Branch

Figure 5: Structure of Complex Queries

Algorithm 1 Compute and Display Overlay

procedure COMPUTEOVERLAY(G,A,S) ▷ G is the graphical tree object; A is the nodes of query expression in DFS order; L is the node ids for one tree returned by query engine; Returns mappings of query nodes to tree nodes
  $LA \leftarrow$ ▷ mapping from L to A
  $c \leftarrow 0$
  for all i in L **do** ▷ for all node ids
    $LA[i] \leftarrow A[c]$
    $c += 1$
  end for
  $LG \leftarrow$ ▷ mapping from L to G
  for all i in L **do** ▷ for all node ids
    $LG[i] \leftarrow G$.search(i)
  end for
  return $LA \bowtie LG$
end procedure
procedure DISPLAYOVERLAY(Q,M) ▷ Q is the query graph; M is the mapping from ComputeOverlay; Updates display
  for all $a(i,j,t)$ in Q **do** ▷ each axis in query
    draw $(M[i],M[j],t)$
  end for
end procedure

5.2 Overlay

The LPath query returns a set of nodes in the database that match the last element in the query. An extension is made to the LPath-to-SQL translator so that all nodes in the query are selected rather than just the last node in the query:

```
select T0.*, T1.* from T T0, T T1
  where T0.type='syn' and T0.name='A'
  and T1.type='syn' and T1.name='B'
  and T0.sid=T1.sid and T0.tid=T1.tid
  and T0.l<=T1.l and T0.r>=T1.r and T0.d<T1.d
```

Each row of the result table returned by this modified SQL query is thus a super-tuple that is a concatenation of sub-tuples, one per node. And in the super-tuple, sub-tuples appear in depth-first-search order relative to the tree representation of the LPath query. Then, for each super-tuple, a list of node ids is extracted, and a mapping from this list to the LPath query tree is computed. Also, using this list of ids, nodes involved in the overlay are identified from the rendered result tree. Finally, axes between nodes are recovered and displayed. These steps are shown in Algorithm 1

6 Prototype

We have implemented a prototype of QBA tool using Python and PyQt. The main components of the tool are described below. Figure 6 provides a screenshot of the tool.

An enriched tree data structure is used to store trees and graphical queries drawn by users. The GUI component renders this structure as a tree, and allows users to annotate it with a graphical query. Once the graphical query is translated to LPath, it is further translated into an SQL query by an LPath-to-SQL translator. We use NLTK to compile the LPath grammar and to parse LPath queries (`nltk.org`, Bird (2006)). In order to support query overlay, the translator also provides a modified SQL translation described in Section 5.2.

The database component maintains a connection to a database server. When a request arrives with an LPath query, it uses the LPath-to-SQL translator to translate it into an SQL query, sends the translated query to the server, and returns the result to the client. A user can connect to either Oracle or PostgreSQL database. Depending on user's connection choice, the database component is configured to one of the database systems at runtime.

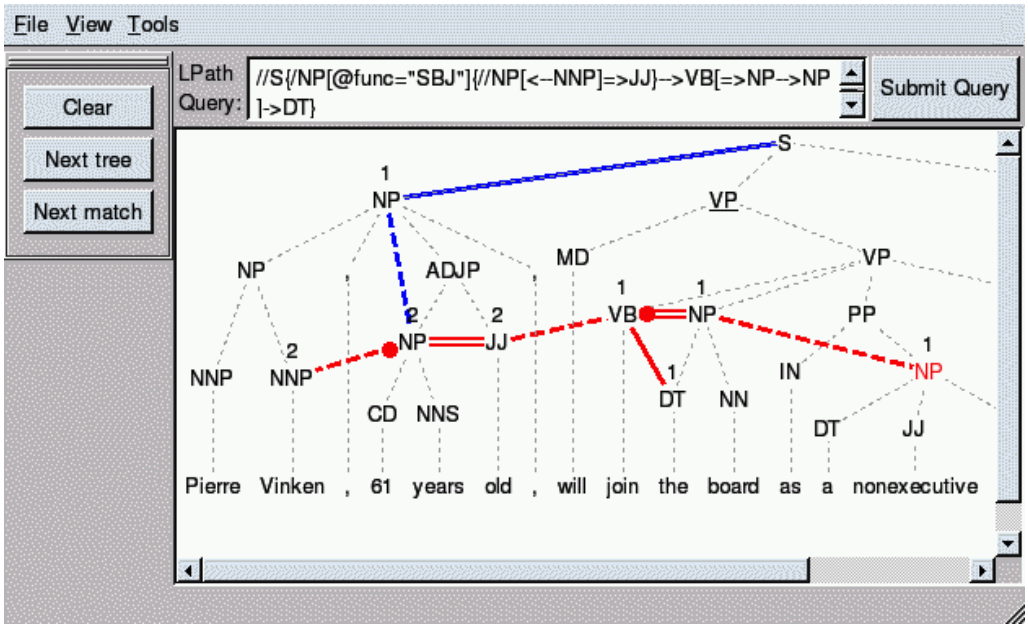


Figure 6: Screenshot of LPath QBA Tool

7 Conclusion

Treebanks have become centrally important in linguistic research and language technology. Treebanks are now being developed for dozens of languages,[1] and for a wider range of linguistic phenomena (e.g. discourse analysis, Miltsakaki et al. (2004)). Earlier work has showed how such tree data can be represented in a relational database and interrogated using a linguistically motivated path language called LPath. Here we have described a new, high-level approach to querying linguistically annotated data, which we call Query by Annotation. Unlike existing approaches to tree query, queries and results are of the same type: annotated trees. Users see why their query matches a result tree. Moreover, they can easily refine their query, constraining or expanding its scope as needed. The resulting approach to graphical query fits naturally into common workflows in linguistic data exploration and curation.

[1] For a list of languages for which treebanks are available, see `http://en.wikipedia.org/wiki/Treebank`

References

Steven Bird. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72. Association for Computational Linguistics, Sydney, Australia. `http://www.aclweb.org/anthology/P/P06/P06-4018`.

Steven Bird, Yi Chen, Susan B. Davidson, Haejoong Lee, and Yifeng Zheng. 2006. Designing and evaluating an XPath dialect for linguistic queries. In *22nd International Conference on Data Engineering*, pages 52–61. `http://eprints.unimelb.edu.au/archive/00001455/`.

Steven Bird and Mark Liberman. 2001. A formal framework for linguistic annotation. *Speech Communication*, 33:23–60. `http://arxiv.org/abs/cs/0010033`.

Daniele Braga, Alessandro Campi, and Stefano Ceri. 2005. XQBE (XQuery By Example): A visual interface to the standard XML query language. *ACM Transactions on Database Systems*, 30(2):398–443.

James Clark and Steve DeRose. 1999. *XML Path language (XPath)*. W3C. `http://www.w3.org/TR/xpath`.

Stephan Kepser. 2003. Finite structure query: a tool for querying syntactically annotated corpora. In *Proceedings of the Tenth Conference*

of the European Chapter of the Association for Computational Linguistics, pages 179–186.

Esther König and Wolfgang Lezius. 2001. The TIGER language: a description language for syntax graphs. part 1: User's guidelines. Technical report, University of Stuttgart, Stuttgart, Germany. `http://citeseer.ist.psu.edu/article/knig01tiger.html`.

Catherine Lai. 2005. *A Formal Framework for Linguistic Tree Query*. Master's thesis, Department of Computer Science and Software Engineering, University of Melbourne.

Catherine Lai and Steven Bird. 2004. Querying and updating treebanks: A critical survey and requirements analysis. In *Proceedings of the Australasian Language Technology Workshop*, pages 139–146. `http://eprints.unimelb.edu.au/archive/00000774/`.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–30. `http://www.cis.upenn.edu/~treebank/home.html`.

Bruno T. Messmer and Horst Bunke. 1998. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The penn discourse treebank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*. Paris: European Language Resources Association. `http://www.seas.upenn.edu/~pdtb/papers/lrec04.pdf`.

Jiří Mírovský. 2006. Netgraph: a tool for searching in prague dependency treebank 2.0. In *Proceedings of The Fifth International Conference on Treebanks and Linguistic Theories*, pages 211–222. `http://quest.ms.mff.cuni.cz/netgraph/pub/2006_tlt.pdf`.

P. Resnik and A. Elkiss. 2003. The linguist's search engine: Getting started guide. Technical Report LAMP-TR-108/CS-TR-4541/UMIACS-TR-2003-109, University of Maryland, College Park. `http://lse.umiacs.umd.edu:8080/`.

D. Rohde. 2001. Tgrep2 user manual. `http://citeseer.ist.psu.edu/569487.html`.

M. M. Zloof. 1977. Query-by-example: A data base language. *IBM Systems Journal*, 16(4):324–343.

Corpus Statistics Approaches to Discriminating Among Near-Synonyms

Mary Gardiner
Centre for Language Technology
Macquarie University
gardiner@ics.mq.edu.au

Mark Dras
Centre for Language Technology
Macquarie University
madras@ics.mq.edu.au

Abstract

Near-synonyms are words that mean approximately the same thing, and which tend to be assigned to the same leaf in ontologies such as WordNet. However, they can differ from each other subtly in both meaning and usage—consider the pair of near-synonyms *frugal* and *stingy*—and therefore choosing the appropriate near-synonym for a given context is not a trivial problem.

Early work on near-synonyms was that of Edmonds (1997). Edmonds reported an experiment attempting to predict which of a set of near-synonyms would be used in a given context using lexical co-occurrence networks. The conclusion of this work was that corpus statistics approaches did not appear to work well for this type of problem and led instead to the development of machine learning approaches over lexical resources such as *Choose the Right Word* (Hayakawa, 1994).

Our hypothesis is that some kind of corpus statistics approach may still be effective in some situations: particularly if the near-synonyms differ in sentiment from each other. Intuition based on work in sentiment analysis suggests that if the distribution of words embodying some characteristic of sentiment can predict the overall sentiment or attitude of a document, perhaps these same words can predict the choice of an individual 'attitudinal' near-synonym given its context, while this is not necessarily true for other types of near-synonym. This would again open up problems involving this type of near-synonym to corpus statistics methods. As a first step, then, we investigate whether attitudinal near-synonyms are more likely to be successfully predicted by a corpus statistics method than other types. In this paper we present a larger-scale experiment based on Edmonds (1997), and show that attitudinal near-synonyms can in fact be predicted more accurately using corpus statistics methods.

1 Introduction

The problem of choosing an appropriate word or phrase from among candidate near-synonyms or paraphrases is important for language generation. Barzilay and Lee (2003) cite summarisation and rewriting as among the possible applications, and point out that a component of the system will need to choose among the candidates based on various criteria including length and sophistication. An application of near-synonym generation is the extension of the text generation system HALogen (Langkilde and Knight, 1998; Langkilde, 2000) to include near-synonyms (Inkpen and Hirst, 2006).

An aspect of the choice between synonyms or paraphrases that should not be neglected is any difference in meaning or attitude. Currently, synonyms and paraphrases are usually treated as completely interchangeable in computational systems. Ideally a system should be able to make a correct choice between *frugal* and *stingy* when trying to describe a person whom the system is intending to praise.

Edmonds (1997) examined a part of this problem: for 7 sets of near-synonyms trying to choose the most 'typical' among them for any given context based on co-occurrence statistics, where typicality is approximated by being able to predict the author's original word choice. This experiment suggested that context was able to predict an author's word choice to an extent. However, while the results improved on the baseline for most

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 31–39,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

cases in the small sample, the results were not considered sufficiently strong to pursue this approach; subsequent work (Inkpen and Hirst, 2006) used machine learning on resources authored by lexicographic experts, such as *Choose the Right Word* (Hayakawa, 1994), to acquire the differences between near-synonyms, although corpus statistics approaches have been used to choose between them (Inkpen et al., 2006).

Very recent work described by Inkpen (2007) has returned to the use of corpus statistics approaches and has discovered that with a sufficiently large amount of training data these approaches are more promising in general. However, neither Edmonds (1997), Edmonds (1999) nor Inkpen (2007) has examined their results in terms of which type of near-synonyms did best, in any case the sample size of 7 was too small to do this.

Differences in nuance between near-synonyms have been categorised in several ways with varying degrees of granularity:

- semantic or denotational variation (*mist* and *fog*) and stylistic or connotational variation (*stingy* and *frugal*) (DiMarco et al., 1993);
- collocational and syntactic variations (*die* and *pass away*), stylistic variations (*house* and *habitation*), expressive variations (*skinny* and *slim*) and denotational variations (*error*, *blunder* and *mistake*) (Edmonds and Hirst, 2002); and
- denotational variations (*invasion* and *incursion*), attitudinal variations (*placid* and *unimaginative*) and stylistic variations (*assistant* and *helper*) (Edmonds and Hirst, 2002; Inkpen and Hirst, 2006).

Sentiment analysis work such as that of Pang et al. (2002) and Turney (2002) suggests that it is possible to acquire the sentiment or orientation of documents using corpus statistics without needing to use lexicographic resources prepared by experts. This also suggests that the sentiment of a word may affect its collocational context quite broadly. For example, taking two cases from the classification scheme above, it seems intuitively plausible that differences between *placid* (positive) and *unimaginative* (negative) may be expressed throughout the document in which they are found, while for the denotational pair *invasion* and *incursion* there is no reason why the document more broadly should reflect the precise propositional differences that are the essence of the denotational subtype. Therefore, it is possible that the results of the Edmond's experiment vary depending on whether the near-synonyms differ in sentiment expressed towards their subject (attitudinal), or whether they differ in some other way.

While the performance of the Edmonds's approach in general is modest, and has factors which may worsen the results including not doing word sense disambiguation, we return to it in this paper in order to test whether the thrust of the approach—using corpus statistics approaches to distinguish between near-synonyms—shows signs of being particularly useful for discriminating among near-synonyms that differ in sentiment. Thus, in this paper we apply Edmonds's approach to a much larger sample of near-synonyms to test whether success varies according to near-synonym type. In Section 2 we outline the near-synonym prediction task. In Section 3 we describe the classification sub-task by which we obtained the data, including an annotation experiment to assess the validity of the classification. In Section 4 we describe the approach to near-synonym prediction, the details of the experiment and its results, along with a discussion. In Section 5 we conclude and present some ideas on what this work might lead on to.

2 Task Description

Edmonds (1997) describes an experiment that he designed to test whether or not co-occurrence statistics are sufficient to predict which word in a set of near-synonyms fills a *lexical gap*. He gives this example of asking the system to choose which of *error*, *mistake* or *oversight* fits into the gap in this sentence:

(1) However, such a move also of cutting deeply into U.S. economic growth, which is why some economists think it would be a big _____.

Edmonds performed this experiment with 7 sets of near-synonyms:

1. the adjectives *difficult*, *hard* and *tough*;
2. the nouns *error*, *mistake* and *oversight*;
3. the nouns *job*, *task* and *duty*;

4. the nouns *responsibility*, *commitment*, *obligation* and *burden*;

5. the nouns *material*, *stuff* and *substance*;

6. the verbs *give*, *provide* and *offer*; and

7. the verbs *settle* and *resolve*.

This small sample size does not allow for any analysis of whether there is any pattern to the different performances of each set and whether or not these differences in performance relate to any particular properties of those sets. Edmonds (1999) repeated the experiment using all of WordNet's synonym sets, but did not break down performance based on any properties of the synsets.

3 Evaluating near-synonym type

3.1 Method

We conducted an annotation experiment to provide a larger test set of near-synonyms to test our hypothesis against. The annotators were asked to decide whether certain WordNet synsets differed from each other mainly in attitude, or whether they differed in some other way.

The synsets were chosen from among the most frequent synsets found in the 1989 Wall Street Journal corpus. We identified the 300 most frequent WordNet 2.0 (Fellbaum, 1998) synsets in the 1989 Wall Street Journal using this frequency function, where $w_1 \ldots w_n$ are the words in the synset and count(w_i) is the number of occurrences of w_i tagged with the correct part of speech in the 1989 Wall Street Journal:

$$\text{(2)} \qquad \text{frequency}_{synset} = \sum_{i=1}^{n} \text{count}(w_i)$$

Synsets were then manually excluded from this set by the authors if they:

1. contained only one word (for example *commercial* with the meaning "of the kind or quality used in commerce");

2. contained a substantial number of words seen in previous, more frequent, synsets (for example the synset consisting of *position* and *place* was eliminated due to the presence of the more frequent synset consisting of *stead*, *position*, *place* and *lieu*);

3. only occurred in a frozen idiom (for example *question* and *head* as in "the subject matter at issue");

4. contained words that were extremely lexically similar to each other (for example, the synset consisting of *ad*, *advertisement*, *advertizement*, *advertising*, *advertizing* and *advert*); or

5. contained purely dialectical variation (*lawyer* and *attorney*).

The aim of this pruning process is to exclude either synsets where there is no choice to be made (synsets that contain a single word); synsets where the results are likely to be very close to that of another synset (synsets that contain many of the same words); synsets where the words in them have very few contexts in which they are interchangable (synsets used in frozen idioms) and synsets where there is likely to be only dialectical or house style reasons for choosing one word over another.

This left 124 synsets of the original 300. These synsets were then independently annotated by the authors of this paper into two distinct sets:

1. synsets that differ primarily in attitude; and

2. synsets that differ primarily in some way other than attitude.

The annotation scheme allowed the annotators to express varying degrees of certainty:

1. that there was *definitely* a difference in attitude;

2. that there was *probably* a difference in attitude;

3. that they were *unsure* if there was a difference in attitude;

4. that there was *probably* not a difference in attitude; or

5. that there was *definitely* not a difference in attitude.

The divisions into *definitely* and *probably* were only to allow a more detailed analysis of performance on the Edmonds experiment subsequent to the annotation experiment. The performance of attitudinal and not-attitudinal sets of synonyms were then compared using the Edmonds methodology.

Table 1: Break-down of categories assigned in the annotation experiment

Difference	Certainty	Annotator		Agreement
		1	2	
Attitude	Definite	14	18	7
	Probable	26	18	9
	Total	40	36	29
Not attitude	Definite	68	63	51
	Probable	15	18	5
	Total	83	81	73
Unsure		1	7	0

Table 2: Inter-annotator agreement and κ scores for the annotation experiment

Category division	κ score	Agreement
Attitudinal, not attitudinal and unable to decide	0.62	82
Annotations where both annotators were sure of their annotation	0.85	97[a]

[a] This figure for inter-annotator agreement is computed by excluding any question which one or both annotators marked as only *probably* belonging to one category or the other, or for which one or both annotators declared themselves unable to decide at all

3.2 Results

Inter-annotator agreement for the annotation experiment is shown in Table 1 both:

- individually for annotations that the annotators felt were *definitely* correct and those that they thought were *probably* correct; and
- collectively, for all annotations regardless of the annotator's certainty.

Two divisions of the annotation results were used to compute a κ score and raw inter-annotator agreement:

1. the agreement between annotators on the "attitudinal difference", "not attitudinal difference" and "unsure" categories regardless of whether they marked their choice as *definite* or *probable*; and
2. the agreement between annotators on *only* the annotations they were definitely sure about, as per Wiebe and Mihalcea (2006).

In fact, we calculated two difference κ scores for each of the above: κ_{Co} assuming different distributions of probabilities among the annotators (Cohen, 1960); and $\kappa_{S\&C}$ assuming identical distributions among the annotators (Siegel et al., 1988) as recommended by Di Eugenio and Glass (2004). However the κ_{Co} and $\kappa_{S\&C}$ values were the same to two significant figures and are thus reported as a single value κ in Table 2. Raw inter-annotator agreement is also shown.

The results suggest we can be fairly confident in using this classification scheme, particularly if restricted to the definite classes.

4 Predicting the most typical word

4.1 Method

In this experiment we replicate the Edmonds (1997) experiment for a larger set of near-synonyms which have been categorised as differing from each other either in attitude or not in attitude, as described in section 3.

Each candidate token c, where a token is a part-of-speech tagged word, such as *(JJ arduous)* or *(NN fight)*, for the gap in sentence S is assigned a score, score(c, S), which is the sum of its significance score with each individual remaining token w in that sentence:

$$\text{score}(c, S) = \sum_{w \in S} \text{sig}(c, w) \quad (3)$$

The candidate c which maximises score(c, S) is chosen as the word fitting the lexical gap in sentence S. Where there is more than one candidate c with an equal maximum value of score(c, S), or where no candidate has a non-zero score, we regard the Edmonds's method as unable to make a prediction.

Edmonds computed the score sig(c, w) by connecting words in a *collocation network*. The principle is that if word w_0 co-occurs significantly with word w_1 which in turn co-occurs significantly with word w_2, then the presence of w_0 should weakly predict the appearance of w_2 even if they do not significantly co-occur in the training corpus. That is, he assumes that if, for example, *task* co-occurs significantly with *difficult*, and *difficult* co-occurs significantly with *learn*, then *task* and *learn* should weakly predict each other's presence.

Edmonds proposes extending this technique to co-occurrence networks with prediction chains of

arbitrary length, but his experimental results suggest that in practice two connections approaches the limit of the usefulness of the technique. Therefore, to compute $\text{sig}(c, w)$ we take the shortest path of significance between the tokens c and w, which is either c, w where c and w significantly co-occur, or c, w_0, w where c and w both significantly co-occur with a third word, w_0.

Where tokens c and w significantly co-occur together, their significance score is their t-score (Church et al., 1991) as calculated by the Ngram Statistics Package (Banerjee and Pedersen, 2003):

(4) $$\text{sig}(c, w) = t(c, w)$$

The t-score is calculated by comparing the likelihood of both words c and w occuring within a certain window of each other. The size of the window is either a 4 word window surrounding c, that is, c and w were found at most 2 words apart, or a 10 word window surrounding c, that is, c and w were found at most 5 words apart.

Where tokens c and w both significantly co-occur with token w_0, their significance score is a combination of their t-scores, with a bias factor devised by Edmonds to account for their weaker connection.

(5) $$\text{sig}(c, w) = \frac{1}{8}(t(c, w_0) + \frac{t(w_0, w)}{2})$$

If there is more than one candidate word w_0 co-occuring significantly with both c and w, the word w_0 is chosen so that the value of $\text{sig}(c, w)$ in equation 5 is maximised.

In the above, we have used "significantly co-occur" without definition. The test we are using is that from the description by Edmonds (1999) of the same experiment: any two words w_0 and w_1 significantly co-occur if their t-scores are greater than 2.0 and their mutual information score is greater than 3.0, as suggested by the observation of Church et al. (1991) that t-scores and mutual information scores emphasise different kinds of co-occurrence.

Input to the t-score and mutual information systems was the part-of-speech tagged 1989 *Wall Street Journal*. Stop words were those used by Edmonds, defined as any token with a raw frequency of over 800 in the corpus, and all punctuation, numbers, symbols and proper nouns. Per Edmonds we did not perform lemmatisation or word sense disambiguation.

As a baseline, also as per Edmonds (1997), we choose the most frequent element of the synset.

4.2 Test synsets and test sentences

Two types of test data were used:

1. lists of WordNet synsets divided into attitudinal and non-attitudinal synsets; and
2. sentences containing words from those synsets.

The lists of synsets is drawn from the annotation experiment described in Section 3. Synsets were chosen where both annotators are certain of their label, and where both annotators have the same label. As shown in Table 1, this results in 58 synsets in total: 7 where the annotators agreed that there was definitely an attitude difference between words in the synset, and 51 where the annotators agreed that there were definitely not attitude differences between the words in the synset.

An example of a synset agreed to have attitudinal differences was:

(6) *bad*, *insecure*, *risky*, *high-risk*, *speculative*

An examples of synsets agreed to not have attitudinal differences was:

(7) *sphere*, *domain*, *area*, *orbit*, *field*, *arena*

The synsets are not used in their entirety, due to the differences in the number of words in each synset (compare {*violence*, *force*} with two members to {*arduous*, *backbreaking*, *grueling*, *gruelling*, *hard*, *heavy*, *laborious*, *punishing*, *toilsome*} with nine, for example). Instead, a certain number n of words are selected from each synset (where $n \in \{3, 4\}$) based on the frequency count in the 1989 Wall Street Journal corpus. For example *hard*, *arduous*, *punishing* and *backbreaking* are the four most frequent words in the {*arduous*, *backbreaking*, *grueling*, *gruelling*, *hard*, *heavy*, *laborious*, *punishing*, *toilsome*} synset, so when $n = 4$ those four words would be selected. When the synset's length is less than or equal to n, for example when $n = 4$ but the synset is {*violence*, *force*}, the entire synset is used.

The sentences for each experiment were selected from one of two corpora: the 1987 Wall Street Journal corpus or the 1988 Wall Street Journal corpus. (Recall that the 1989 Wall Street Journal was used as training data.)

Table 3: Test runs for the Edmonds experiment

Abbreviation	Training window size	Synset size (min 2)	Wall Street Journal year
4win-top3-1987	4	max 3	1987
4win-top4-1987	4	max 4	1987
4win-top4-1988	4	max 4	1988
10win-top3-1987	10	max 3	1987
10win-top4-1987	10	max 4	1987

For each n length synset, all sentences containing one of the words in that synset are found. For example, when the chosen 4-synset is {*hard*, *arduous*, *punishing* and *backbreaking*}, the selected sentences would include:

(8) With closing arguments completed in the first-ever racketeering trial of securities-firm officials, defense lawyers repaired to a downtown bar to celebrate the end of an *arduous* trial.

The word *arduous* is then removed, and the system is asked to predict which of {*hard*, *arduous*, *punishing* and *backbreaking*} goes into the gap using the method described in Section 4.

Table 3 shows the complete test set for the experiment. The same 58 synsets described above are used for all the sets of test data.

- the test sentences may be drawn from the complete 1987 Wall Street Journal, or the complete 1988 Wall Street Journal; and
- the training data is drawn from the 1989 Wall Street Journal, but may have consisted of bigrams drawn from 4 word windows around the target word (the sets called *4win-*), or from bigrams drawn from 10 word windows around the target word (the sets called *10win-*).

4.3 Results and Discussion

Since the Edmonds method cannot always make a prediction, we directly compare the baseline and the Edmonds predictions only on sentences where the Edmond method can make a prediction. The number of times that the Edmonds method can make a prediction at all is shown in Table 4, which also shows the baseline correctness on the sentences described, and the Edmonds method correctness where it can make a prediction. A sentence that contains n words from test synsets is counted as n separate test sentences in this table.

There are several results of interest here. First, the baselines perform noticeably differently for attitudinal versus non-attitudinal success ratios for each of the five data sets. Calculating the z-statistic for comparing two proportions, we find that this difference is significant at the 1% level for each of the data sets, with the attitudinal baseline always higher. Similarly, the difference between attitudinal and non-attitudinal success ratios for Edmonds are also significant at the 1% level.

Because of this first result regarding baselines, the second result, which does show that gross success rates for attitudinal near-synonyms is significantly higher under the Edmonds corpus statistics approach, is less interesting: these higher success ratios could be due to the naturally higher baseline alone.

We inspected some of the data, and noted that for attitudinal synsets, the distribution was much more skewed than for non-attitudinal synsets: one element dominated, and the others were infrequent. In some cases this dominant element appeared to the neutral one, perhaps reflecting the nature of the corpus, but in other cases there was no discernible pattern.

To take into account the varying baselines, we extracted cases where only one method predicted correctly, disregarding those cases where both were right or both wrong. The counts of these are presented in Table 5. We then considered as a 'success' any correct prediction by Edmonds, and calculated the proportion of successes for attitudinal and non-attitudinal for each of the five data sets. Then, for each of the data sets, we compared the success ratios for attitudinal and non-attitudinal, again using the z-statistic as is standard for comparing two proportions (Moore and McCabe, 2003). The differences are again significant at the 1 level. In this analysis, the attitudinal synsets perform better only for 4win-top3-1987 and 10win-top3-1987; that is, for the cases where there are at most three elements in the synset. For the cases with four elements in the synset, the non-attitudinal synsets perform better with respect to the baseline. We speculate that this is due to the nature of the synsets discussed above: the attitu-

Table 4: Performance of the baseline and Edmonds method on all test sentences

	Sentences containing test word		Baseline correctness (%)		Edmonds predictions (%)		Edmonds precision (%)	
Test set	Att.	Non-att.	Att.	Non-att.	Att.	Non-att.	Att.	Non-att.
4win-top3-1987	7588	340246	86.6	66.2	5.7	14.2	94.7	67.5
4win-top4-1987	29453	350038	84.0	65.9	8.1	15.5	72.3	62.9
4win-top4-1988	27023	295437	85.4	64.0	7.7	15.4	69.2	62.0
10win-top3-1987	7588	340246	86.7	67.8	14.7	28.9	90.3	58.7
10win-top4-1987	29453	350038	82.7	67.4	15.2	31.6	65.6	54.1

dinal synsets are distributionally very skewed, and adding a very low probability element (to move from three to four elements in the synset) does not make the task of the baseline noticeably harder, but does add extra noise for Edmonds.

5 Conclusion and Future Work

In this paper we have shown that human annotators can divide near-synonym sets into two classes, those that have members that differ from each other in attitude, and those which do not. We have also investigated whether these two different types of near-synonym sets perform differently when corpus statistics approaches are used to try and discriminate between them.

The data suggests that corpus statistics approaches may perform better on synsets whose members differ in attitude from one another than they do on synsets whose members do not differ in attitude, since the Edmonds methods improves upon the baseline more often when the differences are in attitude than when they are not in attitude.

This preliminary result suggests several further approaches. The first is increasing the size of the size of the word sets used to test any new methods beyond 7 attitudinal words. The second is applying the method of Inkpen (2007) to near-synonyms that differ in attitude rather than that of Edmonds (1997). Edmonds method was the only existing method to use for this comparison at the time of the experiments; but the recent work by Inkpen showed that a larger set of training data greatly improves over Edmonds's original results and suggests that corpus statistics measures are appropriate for discriminating between near-synonyms in general. Her method may confirm whether or near corpus statistics methods apply particularly well to near-synonyms differing in attitude.

Another possibility suggested by a preliminary inspection of the sets of near-synonyms is that attitudinal near-synonyms are distributed differently within the test data. Specifically, the most common word in any attitudinal near-synonym set is particularly common compared to the other words in that set when that word is compared the most common word in a typical non-attitudinal sets. This should be tested and, if confirmed, test data where the word frequencies in attitudinal near-synonym sets more closely match those in non-attitudinal sets may allow for more conclusive tests of the relative performance of corpus statistics measures on the two different types of near-synonym sets.

In the longer term, we hope to use this result to develop new corpus statistics methods to acquire and predict usage of attitudinal near-synonyms, drawing on methods from sentiment analysis.

References

Satanjeev Banerjee and Ted Pedersen. 2003. The design, implementation and use of the Ngram Statistics Package. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*. Mexico City.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *HLT-NAACL 2003: Main Proceedings*, pages 16–23.

Kenneth Church, William Gale, Patrick Hanks, and Donald Hindle. 1991. Using statistics in lexical analysis. In Uri Zernick, editor, *Lexical Acquisition: Using On-line Resources to Build a Lexicon*. Lawrence Erlbaum Associates.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

Barbara Di Eugenio and Michael Glass. 2004. The

Table 5: Number of times each method is right when the baseline and the Edmonds method predict a different word

	All words		Att. words		Non-att. words	
Test data	Baseline	Edmonds	Baseline	Edmonds	Baseline	Edmonds
4win-top3-1987	9715	10824	5	40	9710	10784
4win-top4-1987	13924	12456	604	326	13320	12130
4win-top4-1988	11752	10861	594	256	11158	10605
10win-top3-1987	28900	20825	14	54	28886	20771
10win-top4-1987	37850	23245	1214	449	36636	22796

kappa statistic: a second look. *Computational Linguistics*, 30(1):95–101.

Chrysanne DiMarco, Graeme Hirst, and Manfred Stede. 1993. The semantic and stylistic differentiation of synonyms and near-synonyms. In *Proceedings of AAAI Spring Symposium on Building Lexicons for Machine Translation*, pages 114–121. Stanford, CA, USA.

Philip Edmonds. 1997. Choosing the word most typical in context using a lexical co-occurrence network. 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*, pages 507–509.

Philip Edmonds. 1999. *Semantic representations of near-synonyms for automatic lexical choice*. Ph.D. thesis, University of Toronto.

Philip Edmonds and Graeme Hirst. 2002. Near-synonymy and lexical choice. *Computational Linguistics*, 28(2):105–144.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.

Samuel I. Hayakawa, editor. 1994. *Choose the Right Word*. Harper Collins Publishers. Second Edition, revised by Eugene Ehrlich.

Diana Inkpen. 2007. A statistical model of near-synonym choice. *ACM Transactions of Speech and Language Processing*, 4(1):1–17.

Diana Inkpen and Graeme Hirst. 2006. Building and using a lexical knowledge-base of near-synonym differences. *Computational Linguistics*, 32(2):223–262.

Diana Zaiu Inkpen, Ol'ga Feiguina, and Graeme Hirst. 2006. Generating more-positive or more-negative text. In James G. Shanahan, Yan Qu, and Janyce Wiebe, editors, *Computing Attitude and Affect in Text (Selected papers from the Proceedings of the Workshop on Attitude and Affect in Text, AAAI 2004 Spring Symposium)*, pages 187–196. Springer, Dordrecht, The Netherlands.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics and the 6th Conference on Applied Natural Language Processing (NAACL-ANLP 2000)*, pages 170–177. Seattle, USA.

Irene Langkilde and Kevin Knight. 1998. The practical value of N-grams in generation. In *Proceedings of the 9th International Natural Language Generation Workshop*, pages 248–255. Niagra-on-the-Lake, Canada.

David S. Moore and George P. McCabe. 2003. *Introduction to the Practice of Statistics*. W. H. Freeman and Company, fourth edition.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86.

Sidney Siegel, Castellan, Jr., and N. John. 1988. *Nonparametric statistics for the behavioural sciences*. McGraw Hill, Boston.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 417–424. Philadelphia, Pennsylvania, USA.

Janyce Wiebe and Rada Mihalcea. 2006. Word sense and subjectivity. In *The Proceedings of the The Joint 21st International Conference on Computational Linguistics (COLING) and*

the 44th Annual Meeting of the Association for Computational Linguistics (ACL).

Detecting Compositionality of English Verb-Particle Constructions using Semantic Similarity

Su Nam Kim and Timothy Baldwin
Department of Computer Science and Software Engineering
University of Melbourne, VIC 3010, Australia
{snkim,tim}@csse.unimelb.edu.au

Abstract

We present a novel method for detecting the compositionality of English verb-particle constructions (VPCs), based on the assumption that compositionality can be modelled with semantic similarity between VPCs and their base verb in isolation. We also evaluate the contribution of components in compositional VPCs using semantic overlap.

1 Introduction

There has been a recent surge of interest in the compositionality of multiword expressions (hereafter, MWEs), across a range of languages and constructions (McCarthy et al., 2003; Bannard et al., 2003; Baldwin et al., 2003; Uchiyama et al., 2005; Venkatapathy and Joshi, 2005). For the purposes of this paper, compositionality is defined to be a measure of the degree of correspondence between the semantics of the overall MWE and that of its parts. For example, *put up* (as in *put up the painting*) is compositional as it both involves the action of *putting* and the orientation of *up*. *Make up* (as in *kiss and make up*), on the other hand, doesn't involve *making* in any conventional sense of the word, and the semantic contribution of *up* is less than clear, indicating that it is non-compositional.

In this study, we investigate an automatic method for modelling the compositionality of English verb-particle constructions (VPCs). With VPCs, compositionality relates specifically to the semantic contribution of the head verb and particle—which we will refer to as the *components* of the VPC—as contrasted with the overall VPC. Modelling the compositionality of VPCs has applications in lexicography, in determining whether a VPC is non-compositional and should therefore be lexicalized, with a specialised definition, in the dictionary. Another application is parsing, as compositionality has been shown to impact on word order (Dehé, 2002). Villavicencio (2005) noted that given a compositional VPC, it is often possible to predict novel verb-particle combinations from semantically-homogeneous verbs and particles with similar semantics. This observation has been used to classify particle semantics in VPCs (Cook and Stevenson, 2006), and also provides the motivation for our work. That is, when the verb and particle in a given VPC are similar to those in a VPC which is known to be compositional, it is highly likely that the first VPC is compositional. Hence, we can model the compositionality of unseen VPCs via semantic similarity.

2 Motivation

Villavicencio (2005) claimed that verb–particle combinations share the same or similar semantics when the verbs and particles are similar. To detect the compositionality of VPCs, we extend this observation. When two sets of verb–particle combinations (i.e. VPCs) are semantically the same or similar, then their compositionality is also the same. Hence, by looking at the semantic similarity between unseen VPCs and seen VPCs whose compositionality is already resolved, we can predict the compositionality of unseen VPCs. For example, when *call up* is compositional, we can predict that *ring up* is also compositional due to the similarity between *call* and *ring*.

Figure 1 describes the process of predicting the compositionality of the VPC *sum up*. Here, VPCs are grouped by semantics and each group is tagged with its relative compositionality. For example, *put on* and *take on* in `verb class` *n* are semantically similar. Note that the same VPC can vary in its relative compositionality across different interpretations (c.f. *make up the answer* vs. *kiss and make up*). For the purposes of this research, however, we make the (over-)simplifying assumption

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 40–48,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

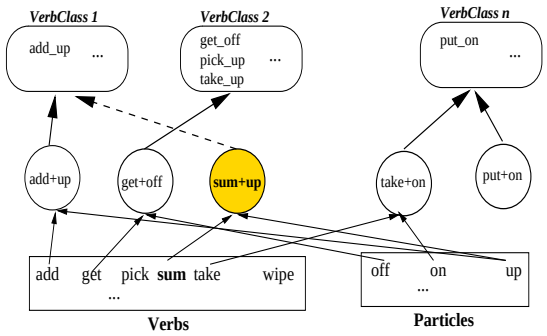


Figure 1: Verb Classes for verb-particle constructions

VPC	compositionality	contribution
lie down	YES	both
close off	YES	verb
get down	YES	particle
make out	NO	none

Table 1: Verb-particle constructions with compositionality and the semantic contribution of its components

that a given VPC has fixed compositionality irrespective of word senses.

In Figure 1, to determine the compositionality of the verb–particle combination *sum up*, we classify it into a semantic class based on lexical similarity. Since *sum* is a synonym of *add*, we predict that *sum up* is in the same semantic class as *add up*, which is compositional. Therefore, we predict that *sum up* is also compositional.

In this process, we need to semantically classify a given combination of verb and particle. To do this, we consider whether each of the verb and particle contribute to the overall semantics of the VPC, leading to four possibilities: verb contribution only, particle contribution only, both verb and particle contribution, or no semantic contribution (similarly to Bannard et al. (2003)).

Examples in Table 1 show the relative semantic contribution of the components in determining the semantics of the overall VPC, and the compositionality of the VPC. As the semantics of the first three VPC examples are derived from the semantics of some or all of its parts, the VPC is compositional. On the other hand, the last example, *make out* is non-compositional since its semantics is not derived from any of its components.

The examples in Table 1 are retrieved from the two available lexical resources: Bannard (2006) and McCarthy et al. (2003). The data is described in detail in Section 4.

Finally, we claim that the semantic similarity of VPCs can be used as a predictor of VPC compositionality. We look at the semantic similarity by checking the semantic contribution of the VPC components. To semantically classify a given combination of verb and particle, we use WORDNET 2.1 hypernyms as the semantic classes of the verb. Note that in our experiment, we use two different forms of hypernyms: direct hypernyms (the first hypernym) and root hypernyms (the *N*th hypernym). For particle semantics, we adopt the classification of Bannard et al. (2003) described in Section 4. To detect the semantic contribution, we build a classifier that takes the semantic class of the VPC, verb and particle as features.

3 Related Work

Lin (1999) used distributional similarity to predict the compositionality of a range of multiword expressions (MWEs), and Schone and Jurafsky (2001) used distributional similarity between the multiword expression candidates and their component words. Both of these works focused primarily on noun–noun MWEs. Baldwin et al. (2003) used a statistical model (in the form of latent semantic analysis) to test the compositionality of VPCs and noun–noun compounds. Detecting the compositionality of VPCs have also been attempted by Bannard et al. (2003) based on the contribution of the verb and particle in a given VPC to the semantics of the VPC. In this work, Bannard et al. (2003) experimented with four tasks: detecting compositionality, one component (either verb or particle) contributes the semantics, verb contributes the semantics, and particle contributes the semantics. McCarthy et al. (2003) also used distributional semantics to predict the compositionality of VPCs, based on the relative similarity between the verb and VPC. McCarthy et al. (2003) modelled similarity via a number of variants of the overlap of the top *N* neighbours (based on distributional similarity). They also carried out comparative evaluation with a thesaurus, man-made resources and collocation measures. Venkatapathy

and Joshi (2005) claimed that if MWEs are decomposable, then they are more likely to be compositional. Piao et al. (2006) tested the compositionality of MWEs using a semantic field taxonomy based on the Lancaster English semantic lexicon.

In this paper, we are exclusively interested in the compositionality of English verb-particle constructions (VPCs). There is a separate but related body of computational linguistic research has been done on the extraction and identification of VPCs. Baldwin and Villavicencio (2002) and Baldwin (2005) extracted VPCs from untagged text using syntactic features. Kim and Baldwin (2006) identified VPCs using an assortment of syntactic and semantic features. O'Hara and Wiebe (2003) proposed a method for preposition sense disambiguation in verb–preposition combinations.

4 Data Collection

We used data from two sources in our experiments: data from Bannard (2006), which individually classifies the compositionality of the verb and particle in VPCs; and data from McCarthy et al. (2003)[1], where the overall VPC is manually scored for compositionality. We describe these two resources in detail below.

4.1 Bannard (2006) Dataset

Bannard (2006) manually tagged 160 VPCs for verb and particle compositionality. The tagging involved 29 human annotators who made a binary judgement for whether first the verb then the particle contributes semantics to the overall VPC. If they could not decide on the contribution of the verb or particle, a tag of "don't know" was assigned. The VPCs in the data set are considered **compositional** if more than half of human annotators answered "yes" for the verb, particle or both. We also generated verb-specific and particle-specific compositionality judgements, giving rise to a total of three datasets: **verb-compositional**, **particle-compositional**, and **VP-compositional**. Table 3 describes the number of compositional and non-compositional VPCs and verbs in isolation in the data set.

Note that in Table 3, the totals are less than the number in the original dataset as a result of filtering out all VPCs where the VPC or base verb was not found in WORDNET 2.1.

[1]Available from `mwe.stanford.edu/resources/`

	VPC	Verb*
Verb-compositional	52	59
Particle-compositional	12	11
VP-compositional	50	62
Non-Compositional	22	25
TOTAL	136	158

Table 2: Breakdown of data from Bannard (2006)

	VPC	Verb*
Compositional	35	58
Non-Compositional	32	39
TOTAL	67	97

Table 3: Breakdown of data from McCarthy et al. (2003)

4.2 McCarthy et al. (2003) Dataset

McCarthy et al. (2003) manually scored 117 VPCs in terms of overall compositionality using a scale of 0 to 10, where 10 indicates the VPC is fully compositional and 0 indicates it is fully non-compositional. For our purposes, we binarised this data such VPCs with a score of 5 or higher are considered to be compositional and all other VPCs are considered to be non-compositional.

Due to the lack of instances in WORDNET 2.1, we finally extracted 67 VPCs and 97 verbs in isolation from McCarthy et al. (2003).

4.3 Semantics of Particles

In order to establish a means to quantify the lexical similarity of particles in VPCs, we classified particles in VPCs according to the following 4-way categorisation (based on Bannard (2003)): TEMPORAL, SPATIAL-DIRECTION, SPATIAL-POSITION and NONE (the last of which is used to tag non-compositional particles). Two human annotators tagged the particles in the VPCs across the two datasets. The initial inter-annotator agreement for this task was 51.71%, and all disagreements were subsequently resolved between the annotators. The final distribution of particles was 86 SPATIAL-DIRECTION, 49 SPATIAL-POSITION and 121 NONE (for a total of 256 VPCs).[2]

[2]Note that there were no instances of TEMPORAL particles in this dataset, as TEMPORAL particles tend to be fully productive and the VPC thus doesn't occur in WORDNET 2.1,

We observed that the semantics of particles do not always match the contribution of the particle in the VPCs. Among the 74 VPCs where the particle was found to be compositional (either particle-compositional or VP-compositional), the particle was tagged as NONE in 22 cases. This discrepancy is due to the fact that we have binarised the compositionality data (by taking the majority decision in the case of the Bannard et al. (2003) data, and applying a threshold of 0.5 for the McCarthy et al. (2003) data), generating a mix of fully and partially compositional VPCs. The partially compositional VPCs cause the problems. For example, in the Bannard et al. (2003) data, 14 annotators marked *knock out* as particle-compositional and 12 annotators marked it as particle-non-compositional. It is thus considered to be particle-compositional for our purposes, but our annotators agreed that the most appropriate semantic class for the particle was NONE.

5 Classifier Construction

We experimented with two approaches to compositionality classification. The first approach (**C1**) models compositionality in the two datasets via the verb and VPC semantics according to WORDNET 2.1, the hand-annotated particle semantics verb–particle co-occurrence data. In the approach, we model compositionality via a verb–particle matrix representation of VPC attestation, based on the research of Uchiyama et al. (2005) over Japanese compound verbs (**C2**). In each case, we experimented with TIMBL (Daelemans et al., 2004) and Zhang Le's MAXENT toolkit.[3]

Uchiyama et al. (2005) successfully performed a Japanese compound verb semantic classification task based on a 2-dimensional matrix representation of which main and auxiliary verb (e.g. *yuderu* "boil" and *ageru* "up") combine to form a compound verb (*yude-ageru* "boil up", in this case). The rows in the matrix represent which auxiliary verb a given main verb combines with to form a compound verb. The feature representation for a given V1-V2 combination is made up of the concatenation of the corresponding row and column in the matrix. Uchiyama et al. (2005) experimented with two basic feature–value representations: (1) simple co-occurrence, i.e. is the given V1-V2 combination attested in corpus data, and (2) semantic categorisation, i.e. is the V1-V2 combination attested in training data, and if so, what is the semantic class associated with it.

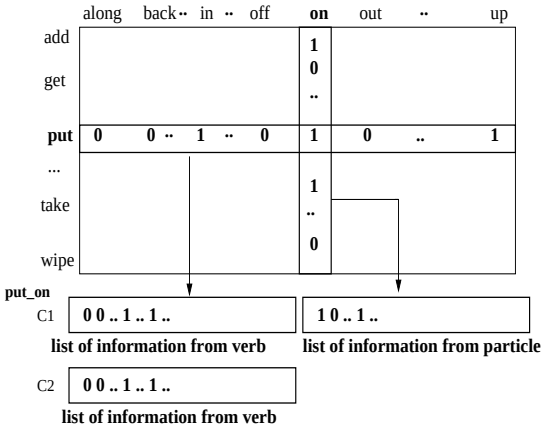


Figure 2: Collocational feature representation

Since the combinatoric properties of English VPCs are similar to those of Japanese compound verbs, this methodology is directly applicable to the compositionality task as targeted in this research. We adapt the two feature–value representations from Uchiyama et al. (2005) in combing up with the following four possibilities: (1) **C–C** where both the verb and particle are represented by way of simple (binary) co-occurrence; (2) **C–S** where the verb is represented via co-occurrence with different particles, but the particle is represented by its semantics (e.g. SPATIAL-POSITION, or 0 if it doesn't produce a VPC in combination with the given verb); (3) **S–C** where the verb is represented via semantics (i.e. WORDNET 2.1 synsets) and the particle via simple co-occurrence; and (4) *S–S* where both the verb and particle are represented via semantics. The input features of two classifiers, (C1) and (C2) are described in Table 4.

The VPC class labels in each case depend on the dataset. For the McCarthy et al. (2003) dataset, the classes are YES and NO, indicating compositionality and non-compositionality, respectively. For the Bannard (2006) dataset, the classes are VERB, PARTICLE, BOTH and NONE.

In Figure 2, (C2) is constructed by concatenating the corresponding row and column of the matrix. The feature–value for each can take the form

e.g. *eat up*. If we removed the restriction on the VPC being lexicalized in WORDNET 2.1, we would expect TEMPORAL VPCs to make up a significant proportion of the data.

[3] `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`

Test	Train	Experiment ID	Classifier Type	Features
VPC	VPC	E1	C1	V_{SEM}, V_{SEM}+P, V_{SEM}+P+P_{SEM}
		E2	C2	C–C, C–S, S–C, S–S
V	V	E3	C1	V_{SEM}, V_{SEM}+P, V_{SEM}+P+P_{SEM}
V	VPC	E4	C1	V_{SEM}, V_{SEM}+P, V_{SEM}+P+P_{SEM}

Table 4: Experimental setup (V_{SEM}= verb semantics, P = particle wordform and P_{SEM}= particle semantics; C = binary co-occurrence and S = semantic class)

Experiment	Features	Semantics	Compositionality	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{F}$
E1	baseline	–	Y	.731	–	–
		–	N	.269	–	–
	V	root	Y	.743	.728	.736
			N	.298	**.315**	.307
E2	baseline	1st, root	Y	.761	–	–
			N	.239	–	–
	S–C	1st	Y	**.791**	.895	**.843**
			N	.429	.250	**.339**
E3	baseline	–	Y	.749	–	–
		–	N	.251	–	–
	V	1st	Y	.771	.864	.818
			N	.366	.234	.300
E4	baseline	–	Y	.598	–	–
		–	N	.402	–	–
	V	1st	Y	.632	**.948**	.759
			N	**.700**	.180	.286

Table 5: Results of experiments with TIMBL (the best result in each column for each of compositional and non-compositional is indicated in **boldface**)

of either simple co-occurrence or semantics, giving rise to the four combinations described in Section 6.1.

6 Evaluation

6.1 Evaluation Set

The underlying hypothesis in this research is that compositionality can be modelled by the lexical semantics of the verb and particle relative to that of the overall VPC. The verb and VPC lexical semantics are represented by WORDNET 2.1 synsets, along with the direct hypernyms and root hypernyms (i.e. unique beginner ancestor) of each. We then represent the semantics of the particles using the 4-way semantic classification described in Section 4.3. Additionally, we have two different classifier structures, (C1) and (C2), based on the simple VPC vs. the collocational properties of the verb and particle in the VPC. As a result, we have 25 different sets of data for the test.

Of the 4 experiments outlined in Table 4 (E1–E4), we used 10-fold cross-validation for E1–E3 and a fixed partitioning of the data for E4, due to the asymmetry in training and test data (verb vs. VPC semantics, respectively).

6.2 Detecting Compositionality

We used two different learners, namely TIMBL and MAXENT, to build our various classifiers. For each experiment, we present a baseline majority-class classifier, along with only the best performing classifier due to space limitations. Evaluation is in terms of precision ($\mathscr{P}$), recall ($\mathscr{R}$) and F-score ($\mathscr{F}$) initially, and then classification accuracy ($\mathscr{A}$) and Pearson correlation (ρ), following McCarthy et al. (2003).

We present the results using TIMBL in Table 5 and those using MAXENT in Table 6. Our classification method performed well when using the semantics of both the VPC and the verb. On the con-

Experiment	Features	Semantics	Compositionality	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{F}$
E1	baseline	–	Y	.731	–	–
		–	N	.269	–	–
	VPS	root	Y	.751	.905	.828
			N	.435	.185	.310
E2	baseline	C–C	Y	.761	–	–
			N	.239	–	–
	C–S	1st	Y	.759	.993	**.876**
			N	.000	.000	.000
E3	baseline	–	Y	.749	–	–
		–	N	.251	–	–
	VPS	1st	Y	**.773**	.895	.836
			N	.429	**.234**	.332
E4	baseline	–	Y	.598	–	–
		–	N	.402	–	–
	VP,VPS	1st	Y	.604	**1.00**	.802
			N	**1.00**	.026	**.513**

Table 6: Results of experiments with MAXENT (the best result in each column for each of compositional and non-compositional is indicated in **boldface**)

trary, while the results are not included in this paper, in general, particle semantics decreased classifier performance due to a lack of data. Also, of the two WORDNET 2.1 sense generalisation methods (1st vs. root hypernyms), we found that 1st hypernyms classified verbs better since they preserve the specific meaning of verbs better.

Comparing the two classifier strategies (C1) and (C2), our results indicate that (C2) produced the best performance among all experiments from both TIMBL and MAXENT, mirroring the results of Uchiyama et al. (2005) for Japanese compound verbs. The highest F-score for TIMBL and MAXENT at detecting compositional VPCs was 0.843 and 0.876, respectively. The results for non-compositional VPC detection were considerably lower, with (C1) producing the highest F-score at 0.513 in experiment 4 using MAXENT. This can be explained by a relative sparsity of non-compositional VPCs in the dataset. Note, the results of all our classifiers for both TIMBL and MAXENT exceeded the baseline in all cases.

6.3 Detecting Semantic Contribution

In this secondary experiment, we test the semantic contribution of the verb and particle in compositional VPCs. The data for this experiment is described in Table 7.

In order to evaluate our approach, we built three classifiers using TIMBL and 10-fold cross-validation: (1) two classifiers based on (C1), with either VPC or verb semantics; and (2) a classifier based on (C2). As shown in Table 8, the highest achieved accuracy of 81.95% was with (C2) using the verb and particle semantics (S–S) and the 1st hypernym of the verb and VPC. Our classifier outperformed the baseline in terms of Precision except for the particle-compositionality task, due to data sparseness.

Feature	Comp	Non-C	total
VPC	112	22	134
	Verb (52)		
	Particle (10)		
	Both (50)		
V	133	25	158
	Verb (59)		
	Particle (62)		
	Both (12)		

Table 7: Number of instances for semantic contribution (Comp = compositionality, Non-C = non-compositionality)

Feature	Comp	Cont	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{F}$
baseline	yes	VP	.373	–	–
		V	.388	–	–
		P	.075	–	–
		All	.836	–	–
	no		.164	–	–
S–S	yes	VP	.489	.460	.475
		V	.474	.745	.610
		P	.000	.000	.000
		All	**.843**	**.964**	**.903**
	no		**.333**	**.091**	**.212**

Table 8: Result of contribution of verb and particle (Comp = compositionality, Cont = contribution)

6.4 Measuring Correlation

Finally, for direct comparison with McCarthy et al. (2003), we evaluate the different classifiers using classification accuracy ($\mathscr{A}$). For the MAXENT classifier, we additionally evaluate the Pearson correlation (ρ) between the compositionality probabilities and the original (non-binarised) human judgements. The results are presented in Table 9. In this case, **boldface** figures indicate above-baseline performance.

The range of correlation statistics reported in McCarthy et al. (2003) was between -0.115 and 0.490 whereas ours were between -0.070 and 0.274. Although the best correlation achieved by our models is lower than that of McCarthy et al. (2003) (0.49 vs. 0.27), it is worth reflecting that McCarthy et al. (2003) used distributional similarity over the entire British National Corpus (i.e. a data and computationally intensive approach), as contrasted with our experiments over small amounts of training data but incorporating lexical semantic resources (i.e. a knowledge intensive approach). Out of our classifiers, (C1) using the direct hypernyms as to represent verb semantics (i.e. E3, with V_{SEM}(1st)) produced the highest correlation.

7 Conclusion

In this study, we proposed a method to automatically detect the compositionality of VPCs based on semantic similarity with their component verb and particle. To evaluate our method, we explored two learning approaches, based on the lexical semantics of only the VPC, verb and particle concerned (C1), and verb and particle co-occurrence (C2). In representing the verb and VPC, we experimented with two methods of semantic generalisation, using direct hypernyms (1st) and the unique beginner ancestor (root). We built classifiers using TIMBL and MAXENT, and evaluated our methods both in terms of precision, recall and F-score for a binary compositionality classification task, and correlation between the marginal probability estimates of the MAXENT model and human judgements (on a continuous scale). The performance of our method exceeded the baseline for each class in terms of F-score. We also evaluated the contribution of the verb and particle in compositional VPCs using the same method to predict the individual verb- and particle-compositionality prediction tasks.

8 Acknowledgement

Many thanks to Dr.Colin Bannard and Dr.Diana McCarthy for data and to the three anonymous reviewers for their valuable input.

References

Timothy Baldwin. 2005. The deep lexical acquisition of English verb-particle constructions. *Computer Speech and Language, Special Issue on Multiword Expressions*, 19(4):398–414.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proc. of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 89–96. Sapporo, Japan.

Timothy Baldwin and Aline Villavicencio. 2002. Extracting the unextractable: A case study on verb-particles. In *Proc. of the 6th Conference on Natural Language Learning (CoNLL-2002)*, pages 98–104. Taipei, Taiwan. URL `http://lingo.stanford.edu/pubs/tbaldwin/conll2002.pdf`.

Colin Bannard. 2003. *Statistical Techniques for Automatically Inferring the Semantics of Verb-Particle Constructions*. Master's thesis, University of Edinburgh.

Colin Bannard. 2006. *Acquiring Phrasal Lexicons*

Experiment	Semantics(N)	TiMBL $\mathscr{A}$	MAXENT $\mathscr{A}$	ρ
E1	baseline	.731		
	V_{SEM}(1st)	.527	.640	.181
	V_{SEM}(root)	.617	.670	.148
	V_{SEM}+P(1st)	.597	.665	.196
	V_{SEM}+P(root)	.617	.680	.148
	V_{SEM}+P+P_{SEM}(1st)	.547	.680	.127
	V_{SEM}+P+P_{SEM}(root)	.592	.715	.085
E2	baseline(C)	.736		
	baseline(1,2)	.751		
	C-C	.711	**.740**	.100
	C-S(1st)	.706	**.755**	-.070
	C-S(root)	.706	**.755**	-.070
	S-C(1st)	.740	.670	.029
	S-C(root)	.735	.665	.020
	S-S(1st)	.685	.720	.155
	S-S(root)	.660	.720	.152
E3	baseline	.749		
	V_{SEM}(1st)	.706	.703	**.274**
	V_{SEM}(root)	.706	.704	.183
	V_{SEM}+P(1st)	.702	.719	.274
	V_{SEM}+P(root)	.671	.692	.163
	V_{SEM}+P+P_{SEM}(1st)	.671	.731	.250
	V_{SEM}+P+P_{SEM}(root)	.667	.700	.140
E4	baseline	.598		
	V_{SEM}(1st)	**.639**	**.630**	.237
	V_{SEM}(root)	**.619**	.580	.072
	V_{SEM}+P(1st)	**.629**	.588	.204
	V_{SEM}+P(root)	**.608**	**.608**	.121
	V_{SEM}+P+P_{SEM}(1st)	.577	**.608**	.157
	V_{SEM}+P+P_{SEM}(root)	**.608**	**.608**	.106

Table 9: Accuracy and correlation (C = the hypernym level, V_{SEM}= verb semantics, P = particle wordform, P_{SEM}= particle semantics)

from Corpora. Ph.D. thesis, University of Edinburgh.

Colin Bannard, Timothy Baldwin, and Alex Lascarides. 2003. A statistical approach to the semantics of verb-particles. In *Proc. of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 65–72. Sapporo, Japan.

Paul Cook and Suzanne Stevenson. 2006. Classifying particle semantics in English verb-particle constructions. In *Proc. of the ACL-2006 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 45–53. Sydney, Australia.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2004. *TiMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide*. ILK Technical Report 04-02. URL `http://ilk.uvt.nl/downloads/pub/papers/ilk0402.pdf`.

Nicole Dehé. 2002. *Particle Verbs in English: Syntax, Information, Structure and Intonation*. John Benjamins, Amsterdam, Netherlands/Philadelphia USA.

Su Nam Kim and Timothy Baldwin. 2006. Automatic extraction of verb-particles using linguistic features. In *11th Conference of European Chapter of the Association for Computational*

Linguistics : Third ACL-SIGSEM workshop on Preposition. Italy.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *In proceedings of ACL-99*, pages 317–324. Maryland.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proc. of the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*. Sapporo, Japan.

Tom O'Hara and Janyce Wiebe. 2003. Preposition semantic classification via Treebank and FrameNet. In *Proc. of the 7th Conference on Natural Language Learning (CoNLL-2003)*, pages 79–86. Edmonton, Canada.

Scott Piao, Paul Rayson, Olga Mudraya, Andrew Wilson, and Roger Garside. 2006. Measuring MWE compositionality using semantic annotation. In *Proc. of the ACL-2006 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 2–11. Sydney, Australia.

Patrick Schone and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *In proceedings of the 2001 conference on Empirical Methods in Natural Language Processing*, pages 100–108. Hong Kong.

Kiyoko Uchiyama, Timothy Baldwin, and Shun Ishizaki. 2005. Disambiguating Japanese compound verbs. *Computer Speech and Language, Special Issue on Multiword Expressions*, 19(4):497–512.

Sriram Venkatapathy and Aravind K. Joshi. 2005. Measuring the relative compositionality of verb-noun (V-N) collocations by integrating features. In *Proc. of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing(HLT/EMNLP 2005)*, pages 899–906. Vancouver, Canada.

Aline Villavicencio. 2005. The availability of verb-particle constructions in lexical resources: How much is enough? *Computer Speech and Language, Special Issue on Multiword Expressions*, 19(4):415–432.

Sentence and Token Splitting Based On Conditional Random Fields

Katrin Tomanek **Joachim Wermter** **Udo Hahn**
Jena University Language & Information Engineering (JULIE) Lab
Fürstengraben 30
D-07743 Jena, Germany
{tomanek|wermter|hahn}@coling-uni-jena.de

Abstract

Natural language processing systems which deal with real-world documents require several low-level tasks such as splitting a text into its constituent sentences, and splitting each sentence into its constituent tokens. These basic text segmentation services are usually supplied by some preprocessor prior to linguistic analysis. While this task is often considered as unsophisticated clerical work, in the natural sciences and engineering domains it poses particular problems due to complex naming conventions. In this paper, we first introduce an annotation framework for sentence and token splitting underlying a newly constructed sentence- and token-tagged biomedical text corpus. This corpus serves as a training environment and test bed for our machine-learning based sentence and token splitters using Conditional Random Fields (CRFs). Our evaluation experiments reveal that CRFs with a rich feature set substantially increase the performance for sentence and token segmentation in scientific documents, *viz.* from the biomedical domain.

1 Introduction

Natural language processing systems expect their input to be properly delimited into consecutive sentences and sentences to be properly segmented into their constituent tokens. Real-world documents lack such fine structure and, typically, simple heuristic pre-processors perform these segmentation tasks. In the newspaper domain (Grefenstette and Tapanainen, 1994), manually defined patterns for sentence splitting rely on the fact that, e.g., a period is a sentence boundary, if it is followed by an uppercase letter and not part of a known abbreviation ('e.g.', 'Mr.', etc.). Tokenization is often performed by splitting at a closed set of special characters (especially punctuation symbols, quotation marks, parentheses, brackets, etc.). While such rules may be adequate for the newspaper domain, they are underspecified when applied to the biomedical domain. Here, complex naming conventions lead, in particular, to unsatisfactory tokenization results when few and simple hand-written rules are used. Both sentence and token boundary symbols are much more ambiguous because they often appear within entity names (organism, protein, cell names, etc.) and their abbreviations, as well as within (chemical) formulae, bibliographic references, etc. (Grover et al., 2006). If these challenges are not accounted for, errors in early processing steps will unavoidably be propagated upwards in an NLP analysis pipeline. As a consequence, text mining modules, such as named entity recognition and relation detection, which are fed by erroneous segmentation results will inevitably suffer in terms of performance.

Rather than adding further complexity to manually maintained rule sets, supervised machine learning (ML) is becoming more and more the method of choice for many demanding NLP tasks. From a given set of training examples a statistical model is automatically learned which is then used to assign labels to unseen data. Those approaches are to a large extent data-driven, i.e., by exchanging the training material they can be ported to other domains and languages, possibly without further changes. This constitutes a clear advantage over rule-based approaches which require a lot of manual tweaking and tuning, especially when more complex rules have to be supplied (Ngai and Yarowsky, 2000). Furthermore, supervised ML approaches have been shown to outperform rule-based ones on several NLP tasks,

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 49–57,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

including sentence boundary detection and tokenization, both in performance and breadth of coverage (Palmer and Hearst, 1997).

Therefore, we developed a new approach to sentence and token splitting based on Conditional Random Fields (CRFs), a sequential machine learning technique (Lafferty et al., 2001). Conditional Random Fields have recently been applied to similar segmentation tasks, such as e.g. word segmentation in Chinese texts (Peng et al., 2004) and sentence boundary detection in speech (Liu et al., 2005). However, to the best of our knowledge, it has not been shown before how CRFs can be applied to sentence splitting and tokenization in scientific documents from the biomedical domain.

For sentence and token boundary detection in the life sciences domain, two annotated text corpora are available which may hold adequate training data for a supervised ML approach, *viz.* the GENIA corpus (Ohta et al., 2002) and the PENNBIOIE corpus (Kulick et al., 2004). Whereas determining the sentence boundaries in a text corpus may be seen as a comparatively easy task, deciding on token boundaries is not as straightforward as it might appear, at first sight. In particular, in the biomedical sublanguages such as evidenced in PUBMED abstracts[1] (or in full articles), crucial semantic units, e.g., entity names or even references to biological processes, can be contained within larger string units and, hence, are not simply delimited by white spaces. A closer look at both corpora reveals that only PENNBIOIE addresses the problem of semantically motivated word token boundary annotation at all, whereas GENIA annotates tokens around the same closed set of special characters as used for the English newspaper language (Marcus et al., 1993).

In this paper, we first report on the compilation and annotation of the JULIE Corpus which provides sentence and word token boundary information in a semantically motivated and linguistically plausible way. The JULIE Corpus is composed of documents from a large variety of biomedical subdomains and entity types with critical word token phenomena. We then report on the JULIE Tools, ML-based tools for sentence and token boundary detection. We evaluate our corpus against GENIA and PENNBIOIE and our tools against another ML-based tool suite, the OpenNLP Tools.[2]

2 Background

To train ML-based tools for sentence and/or token boundary detection, high-quality annotated resources are needed. In particular, several ambiguous non-alphanumeric character symbols may denote sentence or word token boundaries (or not) and, furthermore, these symbols may (or may not) be part of names for biomedical entities, such as protein, cell, or organism names etc. In the following, we outline the annotation guidelines considered for the compilation of the JULIE Corpus which we annotated with sentence and token boundary information.

2.1 Sentence Boundary Annotation

For sentence boundary annotation, it is first necessary to determine potential *sentence boundary symbols* (SBS). For biomedical language texts, such as those from the PUBMED literature database, we defined the 'classical' sentence boundary symbols (“,”, “!”, “?”, “:”) and also two PUBMED-specific ones (“)”, “}”). In particular, for periods (“.”) and colons (“:”) we encountered many cases where they did not denote an SBS:

- General abbreviations (“e.g.”, “i.e.”, “et al.”, “ref.”, “viz.”, “Dr.”, “vs.”, etc.);
- Numbers (“0.05”, “1.2”, “.4”);
- Entity names, e.g., organism names (“E. coli”, “F. oxysporum”, “f. sp. Lycopersicim”, “P. decumbens”). Classifying these symbols as SBS would break up organism names which are essential for disambiguating protein names and mapping them into their database entry (e.g., UniProt). Obviously, simply stating a rule that marks an SBS after each period if the following word starts with a capital letter would over-split the organism entity “f. sp. Lycopersicim”. Moreover, it would also fail to recognize an SBS in cases where the beginning of the following sentence starts with a lower-case letter, as is the case with many protein names (e.g., “p53”, “tac”, etc.);
- Author and journal names in literature citations which are contained in many PubMed

[1] PUBMED is the largest bibliographic database for the life sciences and currently contains more than 16 million entries; see `http://www.ncbi.nlm.nih.gov/entrez/`

[2] `http://opennlp.sourceforge.net`

abstracts (e.g., "Am. J. Physiol.", "J. Biol. Chem.", "L. Hoffmann", "Schindler L.");

- Other alphanumeric strings, such as EC numbers, chromosome locations, database identifiers ("EC1.7.3.3", "LEN.PK113-78");

- Colons followed by enumerations ("Several cytokines interact with each other: IL-2, IL-5, and IL-18.").

2.2 Token Boundary Annotation

For the annotation of word tokens in biomedical text (or any other domain), it is essential to determine which word tokens denote semantic units (i.e., entities) of interest and thus should be recognized as such (e.g., by named entity recognizers). In biomedical text, there are various symbols which may (or may not) denote word token boundary symbols (TBS), such as "-", "+", "/", " ' ", "-", "%", "(", ")", etc., and thus are heavily ambiguous with respect to their status as token boundary symbols. The most important cases are:

- Parentheses must usually be split from regular words, but, e.g., in chemical terminology they are part of the name, such as in "Ca(2+)" or "(S,S)-Tartate". The same holds for enumeration list items such as "1)", "(2)", "a)", "b)", etc.;

- Plus ("+") symbols and hyphens may denote relevant semantic information, such as indicating the presentation (+) of an antigen on a cell or the absence (-) thereof: "CD34(+) T-cells", "CD83+ dendritic cells", "CD11c(++) B-lymphocytes", "CD8alpha(-) DCs". Here, it can be seen that biomedical language expresses complex biological processes by means of a single character symbol. In such cases, of course, these symbols should be tokens on their own;

- Hyphens often concatenate entity names (such as protein names) with other words ("IL-2-specific",[3] "CD28-dependent", etc.) or even with other entity names, such as cell names, as in "CD43-DC". Lacking recognition of these entities as word tokens would prevent entity recognizers from detecting them at all;

- Similar observations can be made with respect to slashes ("/") which often separate two (or more) entity references ("IL-2/CD34", "HA-1/2", etc.);

- Hyphens (and slashes) may also denote the knock-out status of a certain gene with respect to an organism, such as in "flt3L-/- mice".

One might argue that such problems can be overcome by simply splitting (i.e., marking a sentence or token boundary symbol) at every potential split symbol (i.e., at every parenthesis, hyphen, period, colon, etc.). This strategy would, however, split a protein name such as "IL-2" into [IL], [-], and [2]. However, modules further up in a typical text mining pipeline (part-of-speech taggers, phrase chunkers, syntactic parsers, etc.) would not be able to perform adequately on such broken segmentation data because their linguistic representations (either rule-based or derived from training data) could not deal at all with such fragments.

3 Corpus and NLP Tools

3.1 JULIE Sentence and Token Corpus

Currently, only one biomedical text corpus, PENNBIOIE, addresses both sentence boundary and tokenization annotation issues at all, although insufficiently. Furthermore, it is limited to two highly specialized biomedical subdomains, *viz.* the CYP450 enzyme and oncology, focusing on few organism types (human and mouse). No manual token annotation was done on the second well-known annotated biomedical corpus, GENIA, whose domain scope is also rather limited (transcription factors in human blood cells).

Using different sets of MeSH terms[4], we assembled a text corpus consisting of abstracts from PUBMED which covered a more varied set of biological subdomains, including gene expression and regulation, stem cell biology/transplantation and immunology. We refer to this as the *Subdomain Corpus*. Additionally, again by using the MeSH thesaurus, we also assembled subcorpora with respect to biomedical entity types which are known to pose severe problems for sentence and token boundary detection, viz. organisms, chemi-

[3] Here, only the second hyphen should be split because the first one is actually part of the protein name "IL-2".

[4] `http://www.nlm.nih.gov/mesh/meshhome.html`, MeSH is the U.S. National Library of Medicine's controlled vocabulary used for indexing PUBMED articles.

Subdomain + Entity Corpus	20,244
GENIA	18,777
PENNBIOIE	23,358
total	62,379

Table 1: Composition of the JULIE Sentence Corpus (number of sentences)

Subdomain Corpus	12,566
PENNBIOIE	23,358
total	35,924

Table 2: Composition of the JULIE Token Corpus (number of sentences)

cals, cell types and cell components. We refer to this as the *Entity Corpus*.

The Subdomain and Entity Corpora for sentence boundary information were automatically preprocessed with the OpenNLP Sentence Splitter trained on the PENNBIOIE corpus (Buyko et al., 2006). Then the annotations were manually inspected and, if necessary, corrected by a computational linguist and a biologist. In addition, we have added the sentence annotations from GENIA and PENNBIOIE. Altogether, the **JULIE Sentence Corpus** contains 62,379 annotated sentences (see Table 1).

A similar procedure was performed for token annotation. Due to the more complex task, only the Subdomain Corpus was annotated with word token boundary information. Furthermore, we corrected the token annotations in the PENNBIOIE corpus when it did not conform to our guidelines. The **JULIE Token Corpus** contains both the annotated Subdomain Corpus and the corrected PENNBIOIE data, summing up to 35,924 sentences (see Table 2).

3.2 ML-Based JULIE Tools

Our tools for sentence splitting and tokenization are based on supervised machine learning, i.e., from a given set of training examples a statistical model is learned. Such a model is then used to predict labels for unseen data.

Both tasks, i.e., sentence splitting and tokenization, can be considered similar to other chunking tasks where a sequence of (linguistic) units needs to be assigned to labels. This may be interpreted as a traditional classification task where all units of a sequence are considered mutually independent[5] and their labels are thus predicted separately. For the sentence splitter, such a sequence is the complete input text[6] which is decomposed into units of character sequences (split at white space positions). The tokenizer, on the other hand, receives as input sequence a sentence; here, the single units are atomic character sequences which may be combined to tokens.

If, however, there is an inherent interdependency between the single units in a sequence and the independence assumption does not hold, this may lead to a suboptimal classification performance. This is the case in many NLP chunking tasks, including POS tagging, shallow parsing, and especially entity recognition. We assume this to be the case with sentence splitting and tokenization as well. Under the presence of such statistical relational data, the application of sequence models, which assign a sequence of labels to the complete input sequence of units, should be favored due to the natural dependency structure.

Conditional Random Fields (Lafferty et al., 2001) are probabilistic sequence models which have recently been shown to be well suited for different NLP tasks (cf. (Settles, 2004) for named entity recognition, (Sha and Pereira, 2003) for parsing). A Conditional Random Field (CRF) can be interpreted as an undirected graphical model which is globally conditioned on the observation sequence $\mathbf{o}$ of length n and trained so that it maximizes the conditional probability of the label sequence $\mathbf{l} = \langle l_1, ..., l_n \rangle$ and the observation sequence $\mathbf{o} = \langle o_1, ..., o_n \rangle$. A CRF can be arbitrarily structured; however, in most NLP applications, the underlying graph is a simple linear chain. In the following, we will only focus on this case. In a CRF the conditional probability of an observation sequence given a label sequence is thus defined as:

$$P(\mathbf{l}|\mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp\left(\sum_{i=1}^{n} \sum_{k} \lambda_k f_k(l_{i-1}, l_i, \mathbf{o}, i) \right)$$

where $Z(\mathbf{o})$ is a normalization factor which guarantees that we yield a proper probability. A feature function f_k relates the current observation with a bigram of labels; i.e., it is an indicator function associating an observation feature at position

[5] Formally, this is based on the assumption that all these units are independent and identically-distributed examples.

[6] In case of an abstract, the complete abstract may be considered as one sequence. When handling full texts, one might handle each paragraph separately.

i (e.g., *unit at current position i is ","*) and a particular state transition (e.g., *state at position* $i-1$ *is "N" and state at current position i* is "P"). The feature weights λ_k are estimated by maximizing the respective conditional log-likelihood function on the training data by means of iterative-scaling or gradient-based techniques. The higher the value of λ_k, the more important this feature function is.

As an implementation of CRFs, we employed the machine learning toolkit MALLET (McCallum, 2002). In the following, the sentence and the token splitter, referred to as JULIE Tools[7], are explained in detail.

As a general rule, we tried to optimize our tools not only in terms of accuracy, but also with respect to the kind of errors made, *viz.* false positive (FP) and false negative (FN) errors. As for the sentence splitter, we prefer FNs over FPs. As many consecutive NLP components work on the sentence level the costs of erroneously splitting a sentence should be considered higher than not splitting. Here, a false positive means a loss of information which might be especially problematic, e.g., when running an entity tagger or a parser on such a dissected sentence. For the tokenizer, we favor FPs over FNs because in many consecutive processing steps tokens are considered as atomic units; however, if the segmentation is too coarse, again information is lost.

3.2.1 JULIE Sentence Boundary Detector

The input text is broken down into a sequence of (observation) units by splitting at all white space positions. For each such unit, the JULIE Sentence Boundary Detector (JSBD) has to decide whether the end of the sentence has been reached or not. Thus, the following piece of text from a PubMed abstract would be split into the following units (with the actual sentence boundary symbol after "T-cells"):

> ... [on] [IL-2-activated] [CD34(+)] [cytotoxic] [T-cells.] [p3hr-1,] [the] [Burkitt's] [lymphoma] [cell] [line,] [was] ...

Each unit is then represented by the following set of binary features:

- The unit itself (lexical feature) and its size in characters;
- Sentence boundary symbols (SBS):[8] whether the unit ends with SBS, whether the unit contains SBS;
- Canonical word form: transformation rule which replaces capital letters by "A", lowercase letters by "a", digits by "0", and all other characters by "-". Finally identical consecutive letters are collapsed (e.g. IL2→AA0→A0);
- Orthographical features: based on regular expressions (e.g. HasDash, AllCaps, InitalCap, hasParenthesis, ...);
- Abbreviations: whether the unit is contained in the list of known abbreviations, whether the unit conforms to abbreviation classes ("[A-Z].", "([A-Za-z].)+", "[a-z]+.");
- Local context: features of neighboring units in window [-1,1] copied to current unit.

JSBD also employs a rule-based post-processing routine to avoid that a sentence is split within opened parentheses or brackets. This is a useful extension, as scientific papers and abstracts often contain complex bibliographic references within parenthetical or bracket-like expressions. Such a reference should be considered as one sentence. However, within these parenthesized sentences there are often many SBSs, especially periods, which could also be considered as sentence boundaries. Preliminary experiments showed that this processing routine, though it does not improve the overall performance, shifts the FP/FN ratio favorably, i.e., false positives are avoided (clearly at the cost of increasing the rate of false negatives).

3.2.2 JULIE Token Boundary Detector

For the tokenizer we have defined a set of critical *token boundary symbols*[9] (TBS) where possible token boundaries might occur. It should be noted that periods, exclamation and question marks are not included in the TBS set because these are sentence boundary symbols and have been disambiguated by the sentence splitter before. This is in line with the paradigm of running NLP tools in a (sequential) pipeline, where each

[7] A Java implementation of our tools can be downloaded from our website: `http://www.julielab.de`.

[8] Sentence boundary symbols are, e.g., period, question and exclamation mark, etc.

[9] The complete set of token boundary symbols consists of { } , + - ( ) [ ] ; = / & > < '

tool is designated to one NLP task and a (natural) order is implemented for the tools due to inherent input/output dependencies. Thus, tokenization is typically performed after sentence splitting.

At all white space positions and at each TBS we split the sentence into single units, the TBS itself is treated as a separate unit. Our example piece of text from above would thus contain the following units:

> ... [on] [IL] [-] [2] [-] [activated] [CD34] [(] [+] [)] [cytotoxic] [T] [-] [cells] SBS [p3hr] [-] [1] [,] [the] [Burkitt] [’] [s] [lymphoma] [cell] [line] [,] [was] ...

For each such unit, JTBD decides whether this unit constitutes the end of a token or not. A token thus consists of a sequence of n units with the last unit labeled as token end and the other $n-1$ units as inside a token.

Furthermore, we assign each unit to its so-called *super-unit*. So, we also split the sentence into larger strings at each white space position as we did for sentence splitting. Such a string is considered a super-unit of a unit, if it covers this unit in the sentence. In our example piece of text from above, we would construct the following super-units:

> ... [on] [IL-2-activated] [CD34(+)] [cytotoxic] [T-cells.] [p3hr-1,] [the] [Burkitt’s] [lymphoma] [cell] [line,] [was] ...

The super-units are used as context information when we construct the features. Each unit is then represented by the following set of binary features:

- The unit itself, and the super-unit itself (lexical feature);
- Whether the unit had a white space to the right in the original sentence;
- Whether the unit is a TBS;
- Features equivalent to the sentence splitter: size, canonical word form, abbreviation class, local context, rich orthographical features of the unit;
- Bracketing information: whether the super-unit contains brackets (*hasOpeningBracketOnly*, *isInBrackets*, *hasClosingBracketOnly*,...);
- Whether the super-unit is an enumeration (“(1)”), whether the super-unit has genitive (“enzyme’s”), whether the super-unit has plural in brackets (“enzyme(s)”);
- Other orthographical features of the super-unit with focus on hyphens, arrows, +/- symbols which are often contained in biomedical texts, and thus structure chemical names.

4 Experimental Results and Discussion

Sentence splitting and tokenization performance is typically evaluated in terms of the accuracy (A), i.e., the number of correct decisions divided by the total number of decisions being made. Here, the total number of decisions equals the number of units. In our evaluation, we focus on two question, *viz.* (1) how well are the different corpora suited for training, and, (2) whether the JULIE Tools perform better than another well-known ML-based NLP tool suite, the OpenNLP Tools, whose general applicability to the biomedical domain has already been shown (Buyko et al., 2006). Among other NLP tools, OpenNLP also provides a sentence splitter and a tokenizer, both based on conditional maximum entropy models (Ratnaparkhi, 1998), also known as logistic regression. Compared to the rich feature sets of the BioMed Tools, the OpenNLP counterparts have much fewer features (mainly lexical ones) coupled with a non-sequential learning algorithm.

4.1 Sentence Splitter Evaluation

To address the first question, we trained the JULIE Sentence Boundary Detector on both the GENIA corpus (18,529 sentences with approximately 486,000 word tokens) and the original PENNBIOIE corpus (23,277 sentences with approximately 590,000 word tokens). The models learned from this training material were then evaluated against the Subdomain and Entity Corpora. We did not evaluate against the complete JULIE Sentence Corpus because it comprises both the GENIA and the PENNBIOIE corpus. The results are depicted in Table 3. There are only very small differences in the accuracy of the two models (A=99.58 and A=99.62). In addition, we also trained and cross-validated JSBD on the complete JULIE Sentence Corpus which yields an accuracy of A=99.8. Here, we encounter a small improvement over the last two experiments. Table 3 also shows the performance of OpenNLP’s sentence

tool	training data	evaluation data	A	FP
JSBD	GENIA	Subdomain and Entity Corpus	99.58	30%
JSBD	PENNBIOIE	Subdomain and Entity Corpus	99.62	27%
JSBD	10-fold cross-validation on JULIE Sentence Corpus		99.80	30%
OpenNLP	10-fold cross-validation on JULIE Sentence Corpus		98.70	48%

Table 3: Sentence splitter performance: Evaluated on different tools and different corpora

tool	training data	evaluation data	A	FP
JTBD	GENIA	JULIE Token Corpus	71.50	3%
JTBD	PENNBIOIE	JULIE Token Corpus	95.90	25%
JTBD	10-fold cross-validation on JULIE Token Corpus		96.70	45%
OpenNLP	10-fold cross-validation on JULIE Token Corpus		95.00	47%

Table 4: Tokenizer performance: Evaluated on different tools and different corpora

splitter on the complete JULIE Sentence Corpus (10-fold cross-validation). In this setting, JSBD performed significantly better[10] than OpenNLP's sentence splitter (A=99.8 vs. A=98.7).

This result can be explained by the rich feature set in our sentence splitter and the fact that we consider sentence splitting as a sequential learning problem. Also, our sentence splitter produces fewer false positives (FP) than OpenNLP's sentence splitter (FP=30% vs. 48%), which is more favorable for this task.

4.2 Tokenizer Evaluation

We trained the JULIE Token Boundary Detector on GENIA and PENNBIOIE, and evaluated the models on the JULIE Token Corpus to determine how well these corpora are suited for training (see Table 4). As GENIA is only tokenized using newspaper language patterns, it is not so well suited for training; only an accuracy of about A=71.5% is reached. PENNBIOIE is more apt because its word token annotation is more semantically motivated. A 10-fold cross-validation of JTBD on the complete JULIE Token Corpus shows that the performance is thus improved by approximately 1 percentage point. Table 4 also indicates that in comparison with the 10-fold cross-validation of OpenNLP's tokenizer on the complete JULIE Token Corpus (A=95.0), both JTBD's machine learning algorithm (sequential learning) and its rich linguistic feature representation (super-units/units) are superior (A=96.7). Looking at the tokenization decisions, the OpenNLP tokenizer runs into particular problems with hyphens which either are not split at all or, if split, in a rather inconsistent way. Thus, an expression like "IL-2-activated" is sometimes tokenized as [IL-2][-activated], [IL][-][activated], or not at all. Similar errors occur with expressions such as "CD34(+)".

From the above experiments we conclude that for sentence splitting the corpus being used for training is not so critical because sentence annotations are not really controversial within different biomedical subdomains. Tokenization, however, is a much more complex task with respect to the relevant biomedical semantic units to be annotated. Still, performance of tokenization can be significantly improved by employing a tool with a linguistically adequate representation and a rich feature set. In both cases, the extension of an annotation corpus by a set of critical (and rare) subdomain and entity cases, as is done in the JULIE Sentence and Token Corpus, boosts performance.

5 Conclusion

We considered two low-level NLP tasks, *viz.* splitting a text document into its constituent sentences and splitting sentences into their constituent tokens. While these tasks seem to have been sufficiently solved for the general-language newspaper domain, they pose considerable problems for sublanguages in the natural sciences and many engineering domains.

As a solution for this challenge, we propose a sentence splitter and a tokenizer based on Conditional Random Fields, a state-of-the-art sequential machine learning algorithm, that is recently applied to many other NLP tasks. We also argued

[10] On such a level of accuracy, a difference of 1 percentage point is notable because this affects mostly critical cases (organism names, etc.).

for rich(er) feature sets, and built novel corpora for the biomedical domain to train these tools on.

The corpora are basically an extension and a correction of the PENNBIOIE corpus – an extension because we added training material so that this corpus covers a more complete section of the biomedical domain and is not specialized to a subdomain; a correction because we removed some inconsistencies with respect to our annotation guidelines.

Our evaluation experiments run on these corpora indicate that both for the sentence splitting and the tokenization task, a substantial improvement in performance could be achieved. Compared to a maximum entropy approach with poorer feature sets (OpenNLP), these results suggest the superiority of an ML approach based on CRFs and a rich, linguistically-motivated feature set for such NLP preprocessing tasks.

Acknowledgements

This research was funded by the EC's 6th Framework Programme (within the BOOTStrep project under grant FP6-028099), and by the German Ministry of Education and Research (BMBF) within the StemNet project (funding code: 01DS001A to 1C).

References

Ekaterina Buyko, Joachim Wermter, Michael Poprat, and Udo Hahn. 2006. Automatically adapting an NLP core engine to the biology domain. In Hagit Shatkay, Lynette Hirschman, Alfonso Valencia, and Christian Blaschke, editors, *Proceedings of the ISMB 2006 "The Joint Linking Literature, Information and Knowledge for Biology and the 9th Bio-Ontologies Meeting "*. Fortaleza, Brazil, August 2006.

Gregory Grefenstette and Pasi Tapanainen. 1994. What is a word, what is a sentence? Problems of tokenization. In *Proceedings of the 3rd International Conference on Computational Lexicography*, pages 79–87.

Claire Grover, Michael Matthews, and Richard Tobin. 2006. Tools to address the interdependence between tokenisation and standoff annotation. In *Proceedings of the 5th EACL-2006 Workshop on NLP and XML (NLPXML-2006): Multi-Dimensional Markup in Natural Language Processing*, pages 19–26. Trento, Italy, April 4, 2006. East Stroudsburg, PA: Association for Computational Linguistics (ACL).

Seth Kulick, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, Lyle Ungar, Scott Winters, and Pete White. 2004. Integrated annotation for biomedical information extraction. In *Proceedings of the HLT-NAACL 2004 Workshop 'Linking Biological Literature, Ontologies and Databases: Tools for Users – BioLink 2004'*, pages 61–68. Boston, MA, USA, May 2004.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML-2001 – Proceedings of the 18th International Conference on Machine Learning*, pages 282–289. Williams College, MA, USA, June 28 - July 1, 2001. San Francisco, CA: Morgan Kaufmann.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 451–458. Association for Computational Linguistics, Ann Arbor, Michigan.

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.

Andrew K. McCallum. 2002. MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

Grace Ngai and David Yarowsky. 2000. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *ACL'00 – Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 117–125. Hong Kong, China, 1-8 August 2000. San Francisco, CA: Morgan Kaufmann.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In M. Marcus, editor, *HLT 2002 – Human Language Technology Conference. Proceedings of the 2nd International Conference on Human Language Technology Research*, pages 82–86.

San Diego, Cal., USA, March 24-27, 2002. San Francisco, CA: Morgan Kaufmann.

David D. Palmer and Marti A. Hearst. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, 23(2):241–267.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *COLING'04: Proceedings of the 20th International conference on Computational Linguistics*, pages 562–568. Association for Computational Linguistics, Morristown, NJ, USA.

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

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *JNLPBA – Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 107–110. Geneva, Switzerland, August 28-29, 2004.

Fei Sha and Fernando C. N. Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL 2003 – Proceedings of the Human Language Technology Conference and the 3rd Conference of the North American Chapter of the Association for Computational Linguistics*, pages 134–141. May 27 - June 1, 2003, Edmonton, Canada. San Francisco, CA: Morgan Kaufmann.

Using Conditional Random Fields for Clause Splitting

Vinh Van Nguyen **Minh Le Nguyen** **Akira Shimazu**
Japan Advanced Institute of Science and Technology
1-1, Asahidai, Nomi, Ishikawa, 923-1292, JAPAN
{vinhnv, nguyenml, shimazu}@jaist.ac.jp

Abstract

In this paper, we present a Conditional Random Fields (CRFs) framework for the Clause Splitting problem. We adapt the CRFs model to this problem in order to use a very large sets of arbitrary, overlapping and non-independent features. In addition, we propose the use of rich linguistic information along with a new bottom-up dynamic algorithm for decoding to split a sentence into clauses. The experiments show that our result are competitive with the previous results.

1 Introduction

Clauses splitting (CS) is the task of splitting a complex sentence into several clauses. This task is important for various tasks such as machine translation, aligning parallel texts, text to speech systems and transformation from natural language sentences into logical forms and it became shared-task problem in CoNLL 2001 (Sang and Déjean, 2001). CS is a deeper level of partial parsing, which is the task of recovering only a limited amount of syntactic information from natural language processing. There is a lot of interest in the design of learning systems which perform only a partial analysis of a sentence (Abney, 1991; Hammerton et al., 2002).

Recently, many machine learning methods have been successfully applied to partial parsing tasks. Most learning models used for partial parsing tasks are discriminative models. These approaches have reached the-state-of-the-art of partial parsing (Kudo and Matsumoto, 2001; Carreras and Màrquez, 2003; Ando and Zhang, 2005).

Carreras et al. (2005) proposes the discriminative model, so-called FR-Perceptron (Collins, 2002) for recognizing structures of clauses. They used a global online learning algorithm to train a discriminative model for CS. However, FR-Perceptron, which updates the learning functions depending on the predictions on that example. Computation time for the training process is time-consuming in large training sets as described below. This can be inefficient when we apply CS for the other applications.

This paper explores a novel method for CS by exploiting CRFs as an alternative training method to the perceptron. The CRFs model (Lafferty et al., 2001) defines a conditional distribution over labeling given an observation and it allows for the use of very large sets of arbitrary, overlapping and non-independent features and has efficient training and decoding processes which both find globally optimal solution. Roark et al. (2004) claimed that "the CRFs method does a better job of parameter estimation for the same feature set, and is parallelizable, so that each pass over the training set can require just a fraction of the computation time of the perceptron method". Recently, there are many successful applications of CRFs including Shallow parsing (Sha and Pereira, 2003), Named Entity Recognition (McCallum and Li, 2003).

However, using CRFs for CS is not completely simple because there are three weaknesses for applying CRFs:

1. The long distance dependence between a start position (S) and an end position (E) of a clause. It is difficult for the conventional CRFs to deal with the problem of long distance dependence between S and E. We see the example in section 3, which shows the distance from the start position to the end position of the clause is long.

2. Balancing between a number of start word positions and those of end word positions of clauses in a sentence.

3. The clauses can be embedded in the outer

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 58–65,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

clauses.

To overcome the drawbacks mentioned above, we use rich linguistic information and propose a new bottom-up dynamic algorithm for decoding. The experiments show that our results are competitive with the previous results. Especially, the precision of our results significantly improves that of the FR-pereptron method.

The rest of this paper is structured as follows. Section 2 reviews related works. Section 3 formulates the Clause splitting problem. Section 4 briefly introduces linear chain CRFs and how to apply them for Clause Splitting. Section 5 describes and discusses the experimental results. Finally, conclusions are given in Section 6.

2 Related Work

Many supervised methods have been developed for Clause Splitting. Carreras et al. (2005) used a discriminative model approach for it. They applied a global learning algorithm, FR-Perceptron (Collins, 2002) for recognizing structure of clauses. They divided the problem into two layers of local subproblems: a filtering layer, which reduces the search space by identifying plausible clause candidates; and a ranking layer, which builds the optimal clause structure by discriminating among competing clauses. A recognition-based feedback rule is presented, which reflects to each local function its committed error from a global point of view, and follow to train them together online as perceptrons. As a result, the learned function automatically behaves as a filter and ranker, rather than a binary classifier. This is the best result for clause splitting now.

Carreras et al. (2002) applied the Adaboost algorithm as (Carreras and Màrquez, 2001). They improved Clause Identification by using global inference on the top of the outcome hierarchically learned local classifiers. Other approaches such as Maximum Entropy, and Winnow applied for CS too (Hachey, 2002).

A number of different methods for the supervised learning approach were used for the CoNLL-2001 shared task. These were composed of boosting decision trees and decision graphs, neural networks, memory-based learning, statistical, and symbolic learning.

3 Clause Splitting Problem

In a deeper level of partial parsing is clause splitting. A clause is a sequence of words in a sentence and is a grammatical unit that includes, at minimum, a predicate and an explicit or implied subject, and expresses a proposition. The problem is to split a sentence into clauses as follows:

```
(Coach them in (handling complaints) (so
that (they can resolve problems immediately)))
```

The problem is more difficult than simply detecting non-recursive phrases in sentences. Clause Splitting is divided into three parts: identifying clause starts, identifying clause ends, and finding complete clauses (Sang and Déjean, 2001).

Formulation

Let X be a sentence space, and Y be a clause space. We can consider a model for finding clauses as a function $R : X \mapsto Y$ which, given a sentence x, identifies the set of clauses $y \subset Y$ of $x \in X$. First, we assume a filter function F which, given a sentence x consisting of a sequence of n words $(x_1, x_2, \ldots, x_n)$, identifies a set of candidate clauses, $F(x) \subseteq \mathscr{P}$ where $\mathscr{P}$ is the set of all possible clauses. A candidate clause is represented as (s,e) for the sentence x where (s,e) is the sequence of consecutive words from word x_s to word x_e, Second, we assume a *score* function which, given a clause, produces a real-value prediction of the clause. We identify a set of clauses for a sentence according to the following optimality criterion:

$$C(x) = \text{argmax}_{y \subseteq F(x)} \sum_{(s,e)_k \in y} score((s,e)_k, x, y) \qquad (1)$$

in which $C(x)$ is a set of clauses for a sentence x, and $(s,e)_k$ is a *k-th* clause in y.

We will identify the clause starts (task 1) and the clause ends (task 2) to predict a set of candidate clauses for finding complete clauses (task 3).

4 Applying CRFs to Clause Splitting

In this section, we show how to overcome the drawbacks of applying CRFs to CS as mentioned in the introduction. First, we present an overview of the CRFs models, we then propose a decoding algorithm as well as exploiting rich linguistic information to deal with the problem when applying CRFs to CS.

4.1 Conditional Random Fields

Conditional Random Fields (CRFs) (Lafferty et al., 2001) are undirected graphical models used

to calculate the conditional probability of values on designated output nodes given values assigned to other designated input nodes for data sequences. CRFs make a first-order Markov independence assumption among output nodes, and thus correspond to finite state machine (FSMs).

Let $o = (o_1, o_2, \ldots, o_T)$ be some observed input data sequence, such as a sequence of words in a text (values on T input nodes of the graphical model). Let **S** be a finite set of FSM states, each is associated with a label l such as a clause start position. Let $s = (s_1, s_2, \ldots, s_T)$ be some sequences of states (values on T output nodes). CRFs define the conditional probability of a state sequence given an input sequence to be

$$P_\Lambda(s|o) = \frac{1}{Z_o} exp\left(\sum_{t=1}^{T} F(s,o,t)\right) \quad (2)$$

where $Z_o = \sum_s exp\left(\sum_{t=1}^{T} F(s,o,t)\right)$ is a normalization factor over all state sequences. We denote δ to be the Kronecker-δ. Let $F(s,o,t)$ be the sum of CRFs features at time position t:

$$\sum_i \lambda_i f_i(s_{t-1}, s_t, t) + \sum_j \lambda_j g_j(o, s_t, t) \quad (3)$$

where $f_i(s_{t-1}, s_t, t) = \delta(s_{t-1}, l')\delta(s_t, l)$ is an *transition* feature function which represents sequential dependencies by combining the label l' of the previous state s_{t-1} and the label l of the current state s_t, such as the previous label l' = AV (adverb) and the current label l = JJ (adjective). $g_j(o, s_t, t) = \delta(s_t, l)x_k(o,t)$ is a *per-state* feature function which combines the label l of current state s_t and a context predicate, i.e., the binary function $x_k(o,t)$ that captures a particular property of the observation sequence o at time position t. For instance, the current label is JJ and the current word is *"conditional"*.

Training CRFs

Let $\Lambda = \{\lambda_i, \lambda_j\}$ be the set of weights in a CRFs model. Λ is set to maximize the conditional log-likelihood of state sequences in some training set, $D = \{\langle o,s\rangle^{(1)}, \ldots, \langle o,s\rangle^{(N)}\}$:

$$L_\Lambda = \sum_{j=1}^{N} log\left(p_\Lambda(s^{(j)}|o^{(j)})\right) - \sum_k \frac{\lambda_k^2}{2\sigma^2} \quad (4)$$

where the second sum is a Gaussian prior over parameters (with variance σ^2) which provides smoothing to solve with sparsity in the training data.

When the training labels make the state sequence unambiguous, the likelihood function in exponential models such as CRFs is convex, and finding the global optimum is guaranteed. Parameter estimation of a CRFs model requires an iterative procedure. Currently, various methods can be used to optimize L_Λ, including Iterative Scaling algorithms such as GIS and IIS (Lafferty et al., 2001), and quasi-Newton methods such as L-BFGS (Sha and Pereira, 2003). Among these methods, L-BFGS is the most efficient (Malouf, 2002; Sha and Pereira, 2003).

L-BFGS requires only that one provides the first-derivative of the function to be optimized. Let $s^{(j)}$ denote the state path of training sequence j, and then the first-derivative of the log-likelihood is

$$\frac{\delta L_\Lambda}{\delta \lambda_k} = \left(\sum_{j=1}^{N} C_k(s^{(j)}, o^{(j)})\right) - \left(\sum_{j=1}^{N}\sum_s p_\Lambda(s|o^{(j)})C_k(s, o^{(j)})\right) - \frac{\lambda_k}{\sigma^2} \quad (5)$$

in which $C_k(s,o)$ is the count of feature f_k, given s and o. The first two terms correspond to the difference between the empirical and the model expected values of feature f_k. The last term is the first-derivative of the Gaussian prior.

Inference in CRFs

Given the conditional probability of the state sequence defined in (2) and set of the parameters $\Lambda = \{\lambda, \ldots\}$, inference in CRFs is to find the most likely state sequence s^* subject to:

$$s^* = argmax_s p_\Lambda(s|o) = argmax_s \exp\left(\sum_{t=1}^{T} F(s,o,t)\right)$$

We can efficiently calculate s^* with the Viterbi algorithm (Rabiner, 1989). For Viterbi algorithm, we use the table for storing the probability of the most likely path up to time t, which accounts for the first t observations and ends in state s_t . We are given a recursive formulation as follows:

$$\varphi_{t+1}(s_i) = max_{s_j}\{\varphi_t(s_j)\exp(F(s,o,t))\} \quad (6)$$

in which $\varphi_t(s_i)$ $(0 \le t < T-1)$ is the probability of the most likely path up to time t; $\varphi_0(s_i)$ is the probability of starting in each state s_i.

4.2 Features

The set of features we use is similar to that of features reported in (Carreras et al., 2005). The set

Transition feature templates	
Current state: s_t	Previous state: s_{t-1}
l	l'

Per-state feature templates	
Current state: s_t	Context predicate: $x(o,t)$
l	$w_{t-2}; w_{t-1}; w_t; w_{t+1}; w_{t+2}$ $p_{t-2}; p_{t-1}; p_t; p_{t+1}; p_{t+2}$ $c_{t-2}; c_{t-1}; c_t; c_{t+1}; c_{t+2}$ CountBegin(w_n) CountEnd(w_n) where $n \in \{t-2, t-1, t, t+1, t+2\}$

Table 1: Feature templates at word level

of features includes features at word level and features at sentence level.

Features at word level

The features are used with a window representation of size 2. For a window centered at the the word x_t, we use the following features extracted from $(x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2})$. Where x can be:

- Word form (w) and POS tag (p).
- Chunking tag (c).
- Count: the number of a particular linguistic element which appear in a sentence fragment. We consider two fragments of a sentence, with separate features for each: from the beginning of the sentence to w_i (CountBegin), and from w_i to the end (CountEnd). The linguistic elements are enumerated as follows:
  - Relative pronouns (e.g "that", "where", "who", "which", "whom", "whose")
  - Punctuation marks (. , ; :)
  - Quotes
  - Verb phrase chunks
  - Relative phrase chunks

The feature templates at word level is described in Table 1.

Features at sentence level

These features are used for capturing long-distance dependencies and identifying the score function of a clause candidate (s,e):

- Top-most structure: A pattern representing the relevant elements of the top-most structure forming the candidate from s to e. The following elements are used to form the pattern:
  - Punctuation marks
  - Coordinate conjunctions (e.g., "and", "or")
  - The word "that"
  - Relative pronouns (e.g., "that", "which", "who", "whom", "whose")
  - Verb phrase chunks
  - The top clause within the $[x_s, \ldots, x_e]$

  where the pattern only considers the top-most structure. We will ignore a clause which appears in the pattern. In the above example, the pattern for the clause `(Coach them in (handling complaints) (so that (they can resolve problem immediately)))` is VP-%-%-S.
- The number of clauses found inside the candidate $[x_s, \ldots, x_e]$.

4.3 Decoder for Clause Splitting

As mention in section 1 with three weakness, we do not apply CRFs to task 3 directly. In this section, we will describe an algorithm for decoding Clause Splitting in a segment of a sentence from l to r. It is a dynamic algorithm presented in Figure 1 as a recursive function. We use results of task 1 and task 2 as input of task 3. Array $mstart[] = [s_1, s_2, \ldots, s_h]$ and $mend[] = [e_1, e_2, \ldots, e_m]$ store results of task 1 (s_i where $i \in 1, \ldots, h$ is *i-th* start word position of a clause) and task 2 (e_j where $j \in 1, \ldots, m$ is *j-th* end word position of clause) respectively in a segment (l,r); $mstart.size() = h$ and $mend.size() = m$ are the number of mstart array elements and mend array elements respectively. Set $\Gamma = \{s_1, s_2, \ldots, s_h, e_1, e_2, \ldots, e_{m-1}\}$. Bidimensional array $BestClause[l,r]$ stores clauses with an optimal split found in (l,r). Bidimensional array $score[l,r]$ stores the score of the clause candidate (l,r). Recursive function FindClause in Figure 1 includes 4 parameters l, r, i, j, in which i and j are indexes of mstart[] and mend[] respectively. The call to the function **FindClause**$(0, n, 0, mend.size() - 1)$ scans the whole sentence and the optimal clause split for the sentence is stored in BestClause$[0,n]$, in which n is the length of the sentence.

Beginning from line 4 to line 9 of the function uses two recursive calls on the sentence segment to enumerate all clause candidates (s_i, e_j) ($s_i \in$ mstart[], $e_j \in$ mend[]) of segment (l,r). Line 10 of the function finds the optimal split k^* for the

Algorithm 1 FindClause(l: int,r: int,i: int,j: int)

1: **if** $l > r$ **then**
2: **return** 0
3: **end if**
4: **if** $j \geq 1$ **then**
5: FindClause(l, mend[$j-1$], i, $j-1$)
6: **end if**
7: **if** $i < mstart.size() - 1$ **then**
8: FindClause(mstart[$i+1$], r, $i+1$, j)
9: **end if**
10: $k^* = argumax_{k \in \Gamma} score(l,k) + score(k+1,r)$
11: BestClause[l,r]=BestClause[l,k^*]∪BestClause[k^*+1,r]
12: **if** $score(l,r) \geq score(l,k^*) + score(k^*+1,r)$ **then**
13: BestClause[l,r]=BestClause[l,r]∪$\{l,r\}$
14: **end if**

Figure 1: Dynamic Algorithm for Clause Splitting

current sentence segment. The line 11 will assign the union of two disjoint splits BestClause[l, k^*] and BestClause[k^*+1, r] which covers the segment (l,r) to BestClause[l,r] of (l,r) segment. The line 12 and 13 treat the case that a clause (l,r) is added to BestClause[l,r].

A sentence requires a function call for each clause candidate and there is a quadratic number of clause candidates over a number of start words and end words in the sentence. The function consumes a linear time for selecting the optimal split plus the cost of the scoring function. Consequently, computation time of identifying a clause split in a sentence is $O(n^2(n+cost(score)))$ where n is a number of start words and end words in a sentence. Because n is so small, computation time of CS is consumed essentially by computation time of Viterbi algorithm (where $cost(score)$ = computation time of Viterbi algorithm).

Scoring

It is essential that we identify the score of a candidate clause. We use the Viterbi algorithm in the decoding process for task 3. Denote Ω as a set of the outputs which Viterbi algorithm produces to predict boundary labels of clauses in (l,r) segment. The score of a candidate clause (l,r) is defined as follows:

$$\text{score}(l,r) = \sum_{i \in \Omega} \varphi_i(s_k) \tag{7}$$

in which $\varphi_i(s_k)$ is that of (6).

We can smooth the score(l,r) of a candidate clause (l,r) using some linguistic elements of clause candidate (l,r):

- Punctuation marks: n_1
- Coordinate conjunctions (e.g "or", "and"): n_2
- Relative pronouns (e.g "that", "which", "whose", "who", "whom"): n_3
- Verb phrase chunks: n_4

Finally, we define score(l,r) below:

$$\text{score}(l,r) = \sum_{i \in \Omega} \varphi_i(s_k) + \sum_{i=1}^{4} \text{count}(n_i) \tag{8}$$

in which count(n_i) is the number of n_i in clause candidate (l,r).

5 Experiments

We conducted the experiments and evaluated the results with our CRFs framework. We used the Penn Treebank which is used in the CoNLL 2001 shared task [1] (Sang and Déjean, 2001) as data for training and testing the clause splitting. WSJ sections from 15 to 18 were used as training data (8,936 sentences), section 20 as development data (2,012 sentences), and section 21 as test data (1,671 sentences). The data of the CoNLL 2001 shared task includes sentences with words, the clause split solutions, POS labels and chunks labels. The data files contain four column separated by a blank space. Each token (a word or a punctuation mark) is put on a separate line and there is an empty line after each sentence. The first item on each line is a token, the second is the part-of-speech tag of the token, the third is a chunking tag of the token, and the fourth is the named clause label. For task 1, the label of each token defines whether the token is not a start position word of a clause (*), or a start position word of a clause (S). For task 2, the label of each token defines whether the token is not an end position word of a clause (*), or an end position word of a clause (E). For task 3, the label of each token defines whether the token is not a boundary word of a clause (*), or a boundary word of a clause {(S*,*S), ...}. In the example in section 3: `(Coach them in (handling complaints) (so that (they can resolve problem immediately)))`, the named clause labels for task 1: `S * * S * S * S * * * *`, the named clause labels for task 2: `* * * * E * * * * * * E`, the named clause labels for task 3: `(S* * * (S* *S) (S* * (S* * * *`

[1] Data sets are available at http://www.cnts.ua.ac.be/conll2001/clauses/

Word	Tagger	Chunking tag	Start(task1)	End(task2)	Start(task 1)-Chunk	End (task 2)-Chunk
Interactive	JJ	B-NP	S	*	S-B-NP	*-B-NP
Telephone	NN	IN-P	*	*	*-I-NP	*-I-NP
Technology	NN	I-NP	*	*	*-I-NP	*-I-NP
...	...	...	...	...	...	...
Possibilities	NNS	I-NP	*	E	*-I-NP	E-I-NP
.	.	O	*	E	*-O	E-O

Table 2: Integrating Output tag of task 1 and task 2 with chunking tag

`*S)S)S)`. In our system, we used CRF++ (V0.44)[2] to implement the CRFs framework.

We evaluated clause splitting based on the standard measures which are widely used in Information Retrieve (Rijsbergen, 1986): precision (p) - the proportion of correctly recognized clauses in output, recall (r) - the proportion of correctly recognized clauses in correct clauses and their harmonic mean F_1. Let $|\,.\,|$ be the number of elements in a set. The computation of the evaluation in a test set including k elements $\{(x^i, y^i)\}_1^k$ can be formulated below:

$$\mathrm{p} = \frac{\sum_{i=1}^{k} |y^i \cap R(x^i)|}{\sum_{i=1}^{k} |R(x^i)|} \quad \mathrm{r} = \frac{\sum_{i=1}^{k} |y^i \cap R(x^i)|}{\sum_{i=1}^{k} |y^i|} \quad \mathrm{F}_1 = \frac{2pr}{p+r}$$

where $R(x^i)$ is a set of clauses that are identified for a sentence x^i.

For task 1 and task 2, we used the framework CRFs with the set of features in section 4.2 as unigram feature templates. We also used some constraints for Viterbi algorithm in the formula (6) as follows:

- Start position of a clause must be the boundary of a chunk.
- End position of a clause must be the boundary of a chunk.

We combined outputs of task 1 and task 2 with chunking tag respectively to enrich the dependence of its linguistic information. This combining is described in Table 2. The results of task 1 and task 2 are shown in Table 3.

We experimented on task 1 and task 2 with a set of features as bigram feature templates. The results of task 1 and task 2 are shown in Table 4. They show that F1 value of task 1 for test set improves 0.61% and F1 value of task 2 for test set improves 0.94%.

[2]CRF++ is available at http://chasen.org/ taku/software/CRF++/

Category	Precision	Recall	F1
Testa1 (Dev)	96.14%	92.63%	94.35%
Testb1	94.87%	91.56%	93.19%
Testa2 (Dev)	91.31%	88.09%	89.67%
Testb2	90.29%	87.93%	89.09%

Table 3: Task 1 and task 2 results (unigram)

Category	Precision	Recall	F1
Testa1 (Dev)	96.67%	92.85%	94.72%
Testb1	95.23%	92.42%	93.80%
Testa2 (Dev)	92.54%	88.95%	90.71%
Testb2	91.55%	88.56%	90.03%

Table 4: Task 1 and task 2 results (unigram + bigram)

5.1 Using CRFs to predict score

We used CRFs with the bigram feature templates for task 3 presented in Section 4.2. Then we used the formula (6) for Viterbi algorithm to count score(l,r) of clause candidate (l,r) segment. The result of task 3 (identifying the clauses) is shown in Table 5.

5.2 Combining linguistic information

We improved F1 value of task 3 by adding linguistic information for smoothing the score function presented in Section 4.3 and the score function is defined as the formula (8). The result is showed in Table 6. The F1 performance is 84.09%, which improved by 1.25% compared with the case of using the formula (7).

Table 7 shows a comparison of our methods with the previous works on the same training and testing data. The results also show that our method is comparable to that of (Carreras et al., 2005) and outperformed other methods. We see that

Reference	Technique	Precision	Recall	F1
Carreras et al. 05	FR-Perceptron	88.17%	82.10%	85.03%
Our method	**CRFs**	**90.01%**	**78.98%**	**84.09%**
Carreras et al. 02	AdaBoost class	90.18%	78.11%	83.71%
Carreras et al. 01	AdaBoost class	84.82%	78.85%	81.73%
Monila and Pla 01	HMM	70.85%	70.51%	70.68%

Table 7: Comparison of our result and previous results

Category	Precision	Recall	F1
Testa3 (Dev)	87.62%	80.04%	83.66%
Testb3	87.97%	78.45%	82.84%

Table 5: Task 3 results (bigram)

Category	Precision	Recall	F1
Testa3 (Dev)	89.07%	80.30%	84.46%
Testb3	90.01%	78.98%	84.09%

Table 6: Task 3 results (bigram + adding linguistic information)

the result of (Carreras et al., 2005) outperforms our result because they combine the three tasks of CS together. With the error-driven method, they feedback errors on training process. However, our method shows precision significantly improves that of (Carreras et al., 2005). This is very useful when we apply CS for machine translation because the clauses need be identified correctly.

5.3 Relation between performance of task 3 and results of task 1 and task 2

In order to test how the results of task 1 and task 2 effect on the performance of task 3, we conducted an experiment by performing task 3 using the gold standard data of task 1 and task 2. Table 8 shows that F_1 value of the task 3 are 84.28% and 85.99% with adding linguistic information, respectively. We see that the results of using the gold standard data of task1 and task2 (85.99%) could significantly improve our results (84.09%). This explains that the performance of task 1 and task 2 are significantly contributed to the result of splitting clause. However, the main errors of the tasks 3 are the miss-corresponding of the starting point (the result of task1) and the ending point (the result of task 2). These errors are caused by defining the score in decoding algorithm. Our future work is focused on how to find a better scoring method for the decoding algorithm.

6 Conclusion

In this paper, we have presented the CRFs-based framework approach for clause splitting. We have proposed a new bottom-up dynamic algorithm for decoding and some effective linguistic information for clause splitting. We compare the results of exploiting our framework to the previous works in the CONLL 2001 shared task. The experiments have shown our result is competitive with the state-of-the-art results of clause splitting.

In future work, we would like to apply linear programming methods for decoding to constraint some complex features in order to improve the performance of our framework. In other aspect, we would like to use our splitting clause outputs for machine translation to reduce the computational time.

7 Acknowledgments

We would like to thank to anonymous reviewers for helpful discussions and comments on the manuscript. We also would like to thank Taku Kudo for sharing CRF++ toolkit.

The work on this paper was supported by the JAIST 21 century COE program "Verifiable and Evolvable e-Society".

References

Steven Abney. 1991. Parsing by chunks. In *Robert Berwick, Steven Abney, and Caroll Tenney, editors, Principle-Based Parsing*, pages 257–278. Kluwer Academic Publishers, Dordrecht.

Rie Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the Association for Computational Linguistics*, pages 1–9.

X. Carreras and L. Màrquez. 2001. Boosting trees for clause splitting. In *In Proceedings of CoNLL-2001*, pages 73–75. Toulouse, France.

Category	Precision	Recall	F1
Testa3 (Dev) (bigram)	90.34%	80.52%	85.15%
Testa3 (Dev) (bigram + adding linguistic information)	92.96%	80.97%	86.55%
Testb3 (bigram)	90.37%	78.97%	84.28%
Testb3 (bigram + adding linguistic information)	92.19%	80.58%	85.99%

Table 8: Task 3 results (with gold standard result task 1 & task 2)

X. Carreras and L. Màrquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of RANLP-2003*, pages 205–216. Borovets, Bulgaria.

X. Carreras, L. Màrquez, V. Punyakanok, and D. Roth. 2002. Learning and inference for clause identification. In *Proceedings of the 14th European Conference on Machine Learning*, pages 35–47. Finland.

Xavier Carreras, Lluís Màrquez, and Jorge Castro. 2005. Filtering-ranking perceptron learning for partial parsing. *Machine Learning*, 60(1):41–71.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments perceptron algorithms. In *Proceeding EMNLP-02*, pages 1–8. Philadelphia, PA, USA.

Benjamin C. Hachey. 2002. Recognising clauses using symbolic and machine learning approaches. In *Master's thesis*. University of Edinburgh.

J. Hammerton, M. Osborne, S. Armstrong, and W. Daelemans. 2002. Introduction to the special issue on machine learning approaches to shallow parsing. *Journal of Machine Learning Research*, (2):551–558.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of HLT-NAACL01*, pages 192–199. Pittsburgh, USA.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, USA.

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

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. Edmonton, Canada.

L. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. of IEEE*, volume 77, pages 257–286.

Van Rijsbergen. 1986. *Information Retrieval*. Text Book.

B. Roark, M. Saraclar, M. Collins, and M. Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of ACL 04*, pages 47–54.

Erik F. Tjong Kim Sang and Hervé Déjean. 2001. Introduction to the conll-2001 shared task: Clause identification. In *Proceedings of the 5th Conference on Natural Language Learning, CoNLL-2001*, pages 53–57. Toulouse, France.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL03*, pages 213–220.

Incorporating Global Information into Supervised Learning for Chinese Word Segmentation

Hai Zhao and Chunyu Kit
Department of Chinese, Translation and Linguistics,
City University of Hong Kong,
83 Tat Chee Avenue, Kowloon, Hong Kong, China
{haizhao,ctckit}@cityu.edu.hk

Abstract

This paper presents a novel approach to Chinese word segmentation (CWS) that attempts to utilize global information (GI) such as co-occurrence of sub-sequences and outputs of unsupervised segmentation in the whole text for further enhancement of the state-of-the-art performance of conditional random fields (CRF) learning. In the existing work of CWS, supervised and unsupervised learning seldom joined, and thus strengthened, with each other. Our attempt here is to integrate unsupervised learning into supervised learning for CWS. Our experimental results show that character-based CRF framework can effectively make use of global information for performance enhancement on top of the best existing results.

1 Introduction

Machine learning methods have shown their power in Chinese word segmentation (CWS) since the first International Chinese word segmentation Bakeoff in 2003 (Sproat and Emerson, 2003). The task of CWS is to segment an input sequence of characters into a sequence of words. Character-based tagging is a simple but effective formulation of the problem suitable for various competitive supervised machine learning models (Xue, 2003; Peng et al., 2004; Low et al., 2005; Tseng et al., 2005; Zhao et al., 2006a). However, all these models only make use of local information and neglect global information such as the occurrences of a sub-sequence in different sentences. This paper will explore how to integrate such information effectively into supervised learning for CWS.

Conventionally, unsupervised and supervised learning are considered two disjoint categories of main techniques for CWS. The latter relies on a pre-segmented corpus as training data, or at least a predefined lexicon; the former applies when such resources are unavailable (Brent, 1999; Ge et al., 1999; Peng and Schuurmans, 2001; Goldwater et al., 2006). Sophisticated technologies have been developed in both categories. A very interesting question we could not find a good answer in existing work and thus intend to explore in this paper is how the two can join each other effectively, in particular, how the former can be integrated into the latter for performance enhancement.

More specifically, we will integrate two kinds of global information features into our character-based tagging system for CWS and examine their effectiveness. One is whether a sub-sequence occurs in different sequences (namely, sentences), and the other is whether a sub-sequence is identified by unsupervised segmentation as a word. Our experiments show that both of them are effective in improving CRF model's performance on CWS as character-based tagging.

The remainder of the paper is organized as follows. The next section discusses how the two types of global information are extracted from the raw text input in question. Section 3 formulates our integration of such global information into CRF learning for CWS. Then, our experimental results will be presented in Section 4. Section 5 discusses related work on using global information in natural language processing tasks and on ensemble of supervised and unsupervised segmentation. Finally, we summarize our research contribution in Section 6.

2 Unsupervised Segmentation

In principle, unsupervised segmentation assumes no pre-segmented training data nor a pre-defined lexicon.

An unsupervised segmentation strategy has to follow some predefined criterion, e.g., mutual information, to recognize a sub-sequence as a word.

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 66–74,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

(Sproat and Shih, 1990) was an early comprehensive investigation in this direction using mutual information. Many successive works applied mutual information criterion with different ensemble methods (Chien, 1997; Sun et al., 1998; Zhang et al., 2000; Yamamoto and Church, 2001; SUN et al., 2004).

Kit proposed a compression-based unsupervised segmentation algorithm, named after description-length-gain (DLG) based segmentation (Kit and Wilks, 1999; Kit, 2000). This method was used for out-of-vocabulary words identification in (Kit and Liu, 2005).

(Feng et al., 2004) proposed a statistical criterion called *accessor variety* (AV) to measure how likely a sub-sequence is a word, and then to find the best segmentation pattern that maximizes a target function of accessor variety and the length of the sub-sequence as variants. (Jin and Tanaka-Ishii, 2006) proposed *branch entropy* as another criterion for unsupervised segmentation. Both criteria share a similar assumption as in the fundamental work by (Harris, 1970): If the uncertainty of successive tokens increases, then the location is at a border. We consider (Feng et al., 2004) and (Jin and Tanaka-Ishii, 2006) to be, respectively, the discrete and continuous formulation of a similar idea.

The idea behind AV or branch entropy criterion is that word boundary occurs at the point where the uncertainty of successive character increases. Either of these two criterions make use of the measurement of such an uncertainty.

In this paper, we adopt AV as our unsupervised segmentation criterion to find segmented unit candidates, for it handles low-frequent words well, as reported in (Feng et al., 2004). As a measure to evaluate how independent a sub-sequence is, and thus how likely it is a word, the accessor variety of a sub-sequence s of more than one character is defined as

$$AV(s) = min\{L_{av}(s), R_{av}(s)\} \quad (1)$$

where the left and right accessor variety $L_{av}(s)$ and $R_{av}(s)$ are, respectively, defined as the number of distinct predecessor and successor characters. Sub-sequences with an AV value below a given threshold are firstly discarded. The remaining sub-sequences are considered to be potentially meaningful words. In addition, Feng et al. also applied heuristic rules to remove sub-sequences that consist of a word and adhesive characters. In this study, we will drop all additional rules for the sake of simplification.

3 System Description of Supervised Segmentation

3.1 Chinese Word Segmentation via CRF Modeling

Chinese word segmentation (CWS) was first formulated as a character tagging problem in (Xue, 2003), via labeling each character's position in a word. For example, the segmentation,

'自然科学/的/研究/不断/深入',

'natural science / of / research / continuously / deepen',

receives the tag (label) sequence '*BMMESBEBEBE*' as segmentation result, where the four tags B, M, E and S stand, respectively, for the beginning, middle and ending positions in a word, and a single character as a word. A MaxEnt model was trained for such character tagging task in (Xue, 2003).

Conditional random field (CRF) (Lafferty et al., 2001) is a statistical sequence modeling framework that outperforms other popular models such as MaxEnt method. CRF was first applied to CWS in (Peng et al., 2004), treating CWS as a binary decision task for each Chinese character in the input: is it the beginning of a word?

The probability assigned to a label sequence for a particular sequence of characters by a CRF is given by the equation below:

$$P_\lambda(y|s) = \frac{1}{Z}\exp(\sum_{c\in C}\sum_k \lambda_k f_k(y_c, y_{c-1}, s, c)), \quad (2)$$

where $Y = \{y_c\}$ is the label sequence for the sentence, s is the sequence of unsegmented characters, Z is a normalization term, f_k is a feature function, λ_k is the corresponding feature weight, C is the tag set and c indexes into characters in the sequence being labeled. Our implementation of character-based tagging for CWS uses the CRF++ package by Taku Kudo [1].

3.2 Tag Set and Feature Templates of Baseline System

Existing work shows that a 6-tag set enables CRF learning to achieve a better segmentation performance than any other tag sets (Zhao et al., 2006b).

[1] http://chasen.org/~taku/software/CRF++/

Table 1: Definition of 6-tag set

Word Length	1	2	3	4	5	6	7 or longer
Tag Sequence	S	BE	BB_2E	BB_2B_3E	BB_2B_3ME	BB_2B_3MME	$BB_2B_3M...ME$

Table 2: Feature templates for baseline system

Code	Type	Feature	Description
a	Unigram	$C_n, n = -1, 0, 1$	The previous (current, next) character
b	Bigram	$C_nC_{n+1}, n = -1, 0$	The previous (next) character and current character
		$C_{-1}C_1$	The previous character and next character
c	Punctuation, Date, Digital and Letter	$T_{-1}T_0T_1$	T_i is type of previous, current or next character

Thus, we opt for this tag set and corresponding n-gram feature templates as our baseline system.

The 6-tag set includes six tags: B, B_2, B_3, M, E, S. Table 1 illustrates how words with different lengths are tagged with this tag set. Feature templates are shown in Table 2.

We give an explanation to feature template (c). It defines five classes of character sets: numbers or characters whose meanings are numbers represent class 1, those characters whose meanings are date and time represent class 2, English letters represent class 3, punctuation labels represent class 4 and other characters represent class 5.

3.3 Global Feature Templates

The basic idea of using global information for CWS is to inform the supervised learner what sub-sequence is word candidate determined by global information. We do this by marking all characters in the sub-sequences that are identified by either of our criteria, in the same way as we label the pre-segmented training data with the 6-tag set for CRF training. Consider that almost all characters in the text can appear in different sentences and our tagging clique is exactly character, we exclude unigram from global information since character alone brings little useful global information.

As mentioned above, we currently apply two criteria to identify such sub-sequences as likely word candidates, one is whether they co-occur in different sentences and the other is whether they are recognized by AV-based segmentation. Henceforth, we refer to them as COS (Co-Occurrence Sub-sequence) and AVS (Accessor Variety based Sub-sequence), respectively, for brevity.

The next issue we need to determine is the length of such sub-sequences. As for COS, we only use one feature template to express it. Notice that COS could identify too many overlapped sub-sequences. For example, in the following two sentences,

a 菲律宾/总统/埃斯特拉达/宣布/说/，/军方/已经/救出了/11名/人质/。

Philippine / president / Estrada / declared /, /the army / has / rescued / 11 / hostages.

b 埃斯特拉达/二号/接受/一家/电视台/采访时/说/，/三军/参谋长/向/他/报告/人质/已经/获救/。

Estrada / on 2nd / accepted / a / TV channel/ interview / to say /, / the army / chief of staff / reported to / him / the hostages / have been / rescued.

All sub-sequences '埃斯特拉达'(Estrada, the ex-president of Philippine),'埃斯'(Es),'埃斯特'(Est),'埃斯特拉'(Estra), and '斯特拉'(strada) meet the COS criterion. However, only '埃斯特拉达' is the one intended. It is the Longest one. Accordingly, the sub-sequence of tags 'BB_2B_3ME', instead of 'BB_2EBE' or any other tag sub-sequences, is assigned to mark '埃斯特拉达' in both of sentences. Therefore, we will only tag the longest COS sub-sequence in each case for CRF training. To avoid inefficiency in matching and searching for COS sub-sequences with too low reusability elsewhere, we opt for an ad hoc length constraint, that is, only sub-sequences of 2 to 7 characters are considered. According to Zhao et al. (2006b), less than 1% words that are longer than seven-character exist in all kinds of corpora.

The case of AVS is different from COS, in that few sub-sequences identified by AVS overlap each other. Thus, we use different feature templates to represent sub-sequences with different lengths. But, still, for the sake of efficiency, we only consider six feature templates generated by AVS, namely, sub-sequences from bigram to seven-gram.

4 Experimental Results

The experiments are performed in all eight corpora from the second International Chinese Word Segmentation Bakeoff (Bakeoff-2005) and the third International Chinese Language Processing Bakeoff (Bakeoff-2006)[2] (Emerson, 2005; Levow, 2006). Corpus size information is list in Table 3.

Table 3: Corpus size of Bakeoff-2005 and 2006 in number of words

Bakeoff-2005	AS	CityU	MSRA	PKU
Training(M)	5.45	1.46	2.37	1.1
Test(K)	122	41	107	104
Bakeoff-2006	AS	CityU	CTB	MSRA
Training(M)	5.45	1.64	0.5	1.26
Test(K)	91	220	154	100

Word segmentation performance is measured by F-measure ($F = 2RP/(R+P)$), where the recall (R) and precision (P) are the proportions of the correctly segmented words to all words in, respectively, the gold-standard segmentation and a segmenter's output. All experimental results in Subsection 4.1 will be evaluated by F-measure.

COS and AVS features are accumulated from the training corpus and test corpus without any annotation. As for AVS feature, we mark all subsequences of length 2 to 7 for AV value larger than 1.

Our comparison with existing work will be conducted in closed test track of Bakeoff. The rule for the closed test is that no additional information beyond training corpus is allowed, while open test of Bakeoff is without such restrict.

We consider two kinds of baseline results, one achieved only with n-gram feature templates (a) and (b) defined in Table 2, and the other achieved with feature template (c) besides n-gram. [3]

Table 4: Comparisons of the best existing results and ours only with n-gram features in data of Bakeoff-2005

Participant (Site ID)	AS	CityU	MSRA	PKU
Tseng(14)	0.947	**0.943**	**0.964**	0.95
Asahara(15b)	**0.952**	0.941	0.958	0.941
Chen(27)	0.945	0.94	0.960	**0.95**
Best closed	0.952	0.943	0.964	0.95
Kit(33)	0.923	/	0.950	0.916
Zhou(13)	/ /	/ /	0.957 0.966*	0.946
Andrew	/	/	0.968	/
Zhang	0.951	0.951	0.971	0.951
Baseline	0.953	0.948	0.973	0.952
COS	**0.955**	0.954	**0.975**	0.952
AVS	0.954	**0.956**	**0.975**	**0.954**
COS+AVS	0.955	0.956	0.975	0.953
Err. Redu. (%)	4.2	22.8	30.6	6.0

4.1 Comparisons of Best Existing Results and Our Results

The comparison of our segmentation performance against the best in Bakeoff-2005 is presented in Table 4 and 5. All participants with at least a best performance in the closed test of Bakeoff-2005 are given in Table 4 (Asahara et al., 2005; Chen et al., 2005; Tseng et al., 2005). 'Best closed' in this Table means the *official* best results in closed test of Bakeoff-2005.

Some state-of-the-art results after Bakeoff-2005, (Andrew, 2006; ZHOU, 2005; Zhang et al., 2006), are given below the best closed results for further comparison. Note that (ZHOU, 2005) reported an unofficial result (with a star in Table 4) in MSRA corpus. The results of (Kit and Liu,

[2]Bakeoff was an international evaluation proceeding on Chinese word segmentation and named entity recognition held by SIGHAN, a special interest group for Chinese language processing within the association of computational linguistics (ACL). Three Bakeoffs have been held since 2003. Bakeoff-2006 is the latest contest. All corpora used in this study are accessible from the official Bakeoff-2005 and 2006 website, http://www.sighan.org/bakeoff2005 and http://www.sighan.org/bakeoff2006.

[3]As we see, feature template (c) or similar feature templates of character set do cause performance enhancement Low et al. (2005). Some researchers that participated closed test of Bakeoff-2005 and 2006 used such kind of feature templates ZHOU (2005); Tsai et al. (2006); Zhao et al. (2006a); Zhu et al. (2006), while others did not. Therefore, we differentiate between these two baseline results in order to make segmentation results fairly comparable.

2005) is demonstrated for further discussion. In Table 5, the best official results in open test of Bakeoff-2005 are also given.

Table 5: Comparison of the best existing results and ours with n-gram features and features of character set in data of Bakeoff-2005

	AS	CityU	MSRA	PKU
Best closed	0.952	0.943	0.964	0.95
Baseline	0.954	0.956	0.974	0.953
COS	0.958	0.961	**0.975**	0.953
AVS	**0.959**	**0.962**	**0.975**	**0.955**
COS+AVS	0.959	0.962	0.975	0.954
Err. Redu. (%)	14.6	33.3	30.6	8.0
Best open	0.956	0.962	0.972	0.969

A summary of the best results in the closed test of Bakeoff-2006 are presented in Table 6. All participants with at least a third best performance in the closed test of Bakeoff-2006 are given in this table (Carpenter, 2006; Tsai et al., 2006; Wang et al., 2006; Zhao et al., 2006a; Zhu et al., 2006).

Table 6: A summary of the best results in the closed test of Bakeoff-2006

Participant (Site ID)	AS	CityU	CTB	MSRA
Zhu(1)	0.944	0.968	0.927	0.956
Carpenter(9)	0.943	0.961	0.907	0.957
Tsai(15)	0.957	**0.972**	/	0.955
Zhao(20)	**0.958**	0.971	**0.933**	/
Zhang(26)	0.949	0.965	0.926	0.957
Wang(32)	0.953	0.970	0.930	**0.963**
Best closed	0.958	0.972	0.933	0.963

The comparison of our segmentation performance against the best in Bakeoff-2006 is presented in Table 7 and 8. The best official results in open test of Bakeoff-2006 are also given in Table 8.

The rates of error reduction from best closed results of Bakeoff to our method (COS+AVS) are also given in Table 4 through 8. We can observe that our system achieve outstanding performance improvement compared to the best existing results.

Table 7: Comparisons of the best existing results and ours only with n-gram features in data of Bakeoff-2006

	AS	CityU	CTB	MSRA
Best closed	0.958	0.972	0.933	0.963
Baseline	0.954	0.969	0.932	0.961
COS	0.955	0.971	0.938	0.962
AVS	**0.957**	**0.973**	**0.940**	**0.963**
COS+AVS	0.957	0.973	0.940	0.963
Err. Redu. (%)	n/a	3.6	10.4	0.0

Table 8: Comparison of the best existing results and ours with n-gram features and features of character set in data of Bakeoff-2006

	AS	CityU	CTB	MSRA
Best closed	0.958	0.972	0.933	0.963
Baseline	0.959	0.972	0.934	0.961
COS	**0.960**	0.974	0.938	0.963
AVS	**0.960**	**0.975**	**0.941**	**0.963**
COS+AVS	0.960	0.975	0.940	0.963
Err. Redu. (%)	4.8	10.7	10.4	0.0
Best open	0.961	0.977	0.944	0.979

4.2 Discussion

We see that our system demonstrates an significant performance improvement from baseline results and achieves further improvement on top of the state-of-the-art performance for the closed test. More interestingly, it also gives highly comparable results with the best of the open test in Bakeoff, where any extra resources were allowed.

We also find that it is not helpful too much when we attempt to use both COS and AVS features. This potentially suggests that COS and AVS are not independent in feature characteristics.

Since global information is adopted through word candidate information, some researchers may argue that a lexicon that is directly extracted from training corpus or any other linguistic resources can be more helpful for performance enhancement. However, this is not always truth. Existing work did show that a proper external lexicon can be useful for performance improvement (Low et al., 2005), while our empirical study show that lexicon extracted from training corpus will not improve the performance but cause performance loss. The key issue in lexicon usage for CWS is how

to find a good lexicon instead of using a lexicon. Our technique here just defined a method to find a good one, since we observed that it can always cause performance enhancement.

5 Related Work

Global information was shown to be useful in many NLP tasks, especially in named entity recognition (NER). An early work on global information extraction for NER was (Mikheev et al., 1998), via a hybrid system integrating hand-coded rules and machine learning methods. Another attempt at using global information in NER was (Borthwick, 1999), via an additional maximum entropy classifier that tries to correct mistakes by using reference resolution. Chieu and Ng (2002) reported a MaxEnt approach to NER using global features as well as local features, showing better results than Borthwick's reference resolution classifier.

Whole-sentence exponential language models was proposed in (Rosenfeld et al., 2001). Although intended for the modeling of whole sentences, this model can be directly applied to the modeling of whole corpus, and it is quite impressive for its modeling ability in the whole text and label interaction, and it then was adopted, through different formulizations or modifications, in many successive works (Finkel et al., 2005; Takamura et al., 2005; Nakagawa and Matsumoto, 2006).

Our approach of using global information is largely similar to that of (Chieu and Ng, 2002). However, we verify the effectiveness of unsupervised segmentation outputs to supervised segmentation of the training corpus with CRF learning for the first time. In addition, CWS is so primary processing task that it is not trivial to find those analogous features proposed for NER in (Chieu and Ng, 2002) before we consider unsupervised segmentation.

To our knowledge, ensemble of unsupervised and supervised segmentation is a brand-new research area for CWS, in which successful research work has not yet been reported so far. (Kit and Liu, 2005) used a simple divide-and-conquer strategy to integrate unsupervised and supervised segmentation. Their unsupervised segmentation method is DLG-based, while supervised segmentation used example-based learning. DLG was only applied to recognize new words among the sequences of mono-character items in the example-based segmentation output. However, this method was not particularly successful, although the potentials of OOV detection via DLG-based segmentation was illustrated.

The broadly used n-gram language model is not strictly concerned with the use of global information, although it does involve the extraction of global information from the entire training corpus.

(Gao et al., 2005) integrated a trigram model in their log-linear Model for CWS. However, their experimental results were produced in the sense of open test of Bakeoff-2003. Thus it is essentially incomparable between their results and other results for closed test, though our baseline system for closed test can achieve the results as the same level as their system for open test. [4].

(Wang et al., 2006) used a maximum entropy model incorporated with n-gram language model to perform segmentation. Their integration strategy is that the score of a decoding path will be modified by adding the bigram of words with a weight λ at the word boundaries. The modification of path score follows the following formula.

$$V[j,i] = ME[j,i] + min_{i-1}^{k=1}\{V[i-1,k] + \lambda \text{Bigram}(w_{k,i-1}, w_{i,j})\} \quad (3)$$

where $V[j,i]$ is the score of a local best path which ends at the jth character, $ME[j,i]$ is the score output from the MaxEnt tagger, and the last word on the path is $w_{i,j} = c_i...c_j$, the weight parameter λ is optimized by the test set used in Bakeoff-2005. Note that ensemble parameters in Wang et al. (2006) were acquired from additional linguistic resource[5]. In contrast, our approach here start from an simple assumption, i.e., accessor variety, as a criteria of unsupervised segmentation in training corpus without annotation.

(Zhang et al., 2006) reported an ensemble learning method for CWS that integrated n-gram language model into a sub-word based tagging system. However, their n-gram language model was

[4]We do not use Bakeoff-2003 data sets for our experiments. In fact, our baseline system only with n-gram features could achieve F-scores 0.973,0.948,0.873 and 0.956 respectively for the four Bakeoff-2003 corpora, AS, CityU, CTB, and PKU, while the corresponding F-scores in Gao et al. (2005) (for open test) are 0.958, 0.954, 0.904 and 0.955, respectively.

[5]This violates the rule of closed test in the Bakeoff that nothing else than the training corpus was allowed for training. Consider the fact that MSRA2006 training corpus is a subset of MSRA2005 training corpus. This may explain why their result is so much better than that of second best.

trained with an annotated corpus instead of plain texts and intended to recognize known words. They adopted a weighting scheme as the basic strategy for integrating n-gram models. A confidence measure was calculated from joint probability of sub-word tagging procedure: If it is higher than an empirical threshold, then the sub-word tagging were applied; otherwise, stay with the dictionary-based n-gram model.

Comparison between these existing works and ours is given in Subsection 4.1, showing that our system achieves better results, in general.

In practice, CRF character-based tagging for CWS can be a heavy computation burden for many current hardware settings. This situation could become worse if more tags were introduced for the purpose of performance enhancement. Thus, a proper tradeoff between computational efficiency and performance is also an important issue with this learning framework. Our experience tells that the proposed method integrated with global information adds only some minor computational cost beyond the baseline, especially for COS feature generation.

6 Conclusion

In this paper, we have presented a novel approach to integrating global information such as outputs of unsupervised segmentation to supervised learning for Chinese word segmentation. We learnt no previous work on CWS that ever attempted to strengthen supervised learning with unsupervised learning outcomes. We provide evidence to show that character-based CRF modeling for CWS can make use of global information effectively, and accordingly achieve a performance better than the best records in the past, according to our experimental results with the latest Bakeoff data sets.

This ensemble strategy allows global features to be used in exactly the same way as local features. In this regard, our approach is straightforward, easy for implementation, and highly adaptable, besides its effectiveness and efficiency.

Acknowledgements

The research described in this paper was supported by the Research Grants Council of Hong Kong S.A.R., China, through the CERG grant 9040861 (CityU 1318/03H) and by City University of Hong Kong through the Strategic Research Grant 7002037. Dr. Hai Zhao was supported by a postdoctoral Research Fellowship in the Department of Chinese, Translation and Linguistics, City University of Hong Kong.

References

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 465–472. Association for Computational Linguistics, Sydney, Australia.

Masayuki Asahara, Kenta Fukuoka, Ai Azuma, Chooi-Ling Goh, Yotaro Watanabe, Yuji Matsumoto, and Takashi Tsuzuki. 2005. Combination of machine learning methods for optimum Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 134–137. Jeju Island, Korea.

Andrew Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, Computer Science Department, New York University.

Michael R. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

Bob Carpenter. 2006. Character language models for Chinese word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 169–172. Association for Computational Linguistics, Sydney, Australia.

Aitao Chen, Yiping Zhou, Anne Zhang, and Gordon Sun. 2005. Unigram language model for Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 138–141. Jeju Island, Korea.

Lee-Feng Chien. 1997. PAT-tree-based keyword extraction for Chinese information retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–58. Philadelphia.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: A maximum entropy approach using global information. In *Proceedings of the 19th International Conference*

on Computational Linguistics (COLING 2002), page 190 – 196. Taipei, Taiwan.

Thomas Emerson. 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123–133. Jeju Island, Korea.

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. 2004. Accessor variety criteria for Chinese word extraction. *Computational Linguistics*, 30(1):75–93.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, page 363 – 370. Association for Computational Linguistics, Ann Arbor, Michigan.

Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computational Linguistics*, 31(4):531–574.

Xianping Ge, Wanda Pratt, and Padhraic Smyth. 1999. Discovering Chinese words from unsegmented text. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 271–272. ACM, Berkeley, CA, USA.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *COLING/ACL 2006*, pages 673–670. Sidney, Australia.

Zellig Sabbetai Harris. 1970. Morpheme boundaries within words. In *Papers in Structural and Transformational Linguistics*, page 68 – 77.

Zhihui Jin and Kumiko Tanaka-Ishii. 2006. Unsupervised segmentation of Chinese text by use of branching entropy. In *COLING/ACL 2006*, pages 428–435. Sidney, Australia.

Chunyu Kit. 2000. *Unsupervised Lexical Learning as Inductive Inference*. Ph.D. thesis, University of Sheffield.

Chunyu Kit and Xiaoyue Liu. 2005. An example-based Chinese word segmentation system for CWSB-2. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 146–149. Jeju Island, Korea.

Chunyu Kit and Yorick Wilks. 1999. Unsupervised learning of word boundary with description length gain. In M. Osborne and E. T. K. Sang, editors, *CoNLL-99*, pages 1–6. Bergen, Norway.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117. Sydney, Australia.

Jin Kiat Low, Hwee Tou Ng, and Wenyuan Guo. 2005. A maximum entropy approach to Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 161–164. Jeju Island, Korea.

Andrei Mikheev, Claire Grover, and Marc Moens. 1998. Description of the LTG system used for MUC-7. In Nancy A. Chinchor, editor, *Proceedings of the Seventh Message Understanding Conference(MUC-7)*. Fairfax, Virginia.

Tetsuji Nakagawa and Yuji Matsumoto. 2006. Guessing parts-of-speech of unknownwords using global information. In *COLING/ACL 2006*, pages 705–712. Sidney, Australia.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *COLING 2004*, pages 562–568. Geneva, Switzerland.

Fuchun Peng and Dale Schuurmans. 2001. Self-supervised Chinese word segmentation. In *The 4th International Symposium on Intelligent Data Analysis*, pages 238–247. Lisbon, Portugal.

Ronald Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. 2001. Whole-sentence exponential

language models: A vehicle for linguistic-statistical integration. *Computers Speech and Language*, 15(1):55 – 73.

Richard Sproat and Thomas Emerson. 2003. The first international Chinese word segmentation bakeoff. In *The Second SIGHAN Workshop on Chinese Language Processing*, pages 133–143. Sapporo, Japan.

Richard Sproat and Chilin Shih. 1990. A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese and Oriental Languages*, 4(4):336–351.

Mao Song SUN, Ming XIAO, and Benjamin K. Tsou. 2004. Chinese word segmentation without using dictionary based on unsupervised learning strategy (in Chinese) (基于无指导学习策略的无词表条件下的汉语自动分词). *Chinese Journal of Computers*, 27(6):736–742.

Maosong Sun, Dayang Shen, and Benjamin K. Tsou. 1998. Chinese word segmentation without using lexicon and hand-crafted training data. In *COLING-ACL '98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, volume 2, pages 1265–1271. Montreal, Quebec, Canada.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, page 133 – 140. Association for Computational Linguistics, Ann Arbor, Michigan.

Richard Tzong-Han Tsai, Hsieh-Chuan Hung, Cheng-Lung Sung, Hong-Jie Dai, and Wen-Lian Hsu. 2006. On closed task of Chinese word segmentation: An improved CRF model coupled with character clustering and automatically generated template matching. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117. Sydney, Australia.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for SIGHAN bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168–171. Jeju Island, Korea.

Xinhao Wang, Xiaojun Lin, Dianhai Yu, Hao Tian, and Xihong Wu. 2006. Chinese word segmentation with maximum entropy and N-gram language model. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 138–141. Sydney, Australia.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Mikio Yamamoto and Kenneth W. Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27(1):1–30.

Jian Zhang, Jianfeng Gao, and Ming Zhou. 2000. Extraction of Chinese compound words – an experimental study on a very large corpus. In *Proceedings of the Second Chinese Language Processing Workshop*, pages 132–139. Hong Kong, China.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging for confidence dependent Chinese word segmentation. In *COLING/ACL 2006*, pages 961–968. Sidney, Australia.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006a. An improved Chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 162–165. Sydney, Australia.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006b. Effective tag set selection in Chinese word segmentation via conditional random field modeling. In *Proceedings of PACLIC-20*, pages 87–94. Wuhan, China.

Guo Dong ZHOU. 2005. A chunking strategy towards unknown word detection in Chinese word segmentation. In Robert Dale, Kam-Fai Wong, Jian Su, and Oi Yee Kwong, editors, *IJCNLP 2005*, volume 3651 of *Lecture Notes in Computer Science*, pages 530–541. Jeju Island, Korea.

Mu-Hua Zhu, Yi-Lin Wang, Zhen-Xing Wang, Hui-Zhen Wang, and Jing-Bo Zhu. 2006. Designing special post-processing rules for SVM-based Chinese word segmentation. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 217–220. Sydney, Australia.

Combining Similarity-based Approaches for Semantic Tagging of Unknown Words

Fumiyo Fukumoto
Interdisciplinary Graduate School of Medicine and Engineering
Univ. of Yamanashi
fukumoto@yamanashi.ac.jp

Yoshimi Suzuki
Interdisciplinary Graduate School of Medicine and Engineering
Univ. of Yamanashi
ysuzuki@yamanashi.ac.jp

Abstract

This paper presents a method for semantic classification of unknown words which are not described in the Japanese thesaurus dictionary, called Bunrui-Goi-Hyo(BGH). The method combines three approaches based on similarity measure: (1) edit-distance, (2) *k*-th Reciprocal Nearest Neighbors(RNN), and (3) semi-supervised clustering, each of them is different in the aspect of unknown words. The final output is obtained by voting. Despite the simplicity of the method, empirical results show that the method is effective, and captures more delicate meaning of words.

1 Introduction

Many semantic-oriented applications such as Question Answering, Paraphrasing, and Machine Translation systems need to recognize which words may substitute each other in a meaning preserving manner(D.Lin, 2001), (B.Regina and K.McKeown, 2001), (S.Ikehara et al., 1994). For example, we can accept that two sentences 'He could not find serious *shortcomings*' and 'He could not find serious *failing*' are almost the same sense. Fine-grained and large-scale semantic knowledge like WordNet, COMLEX, EDR dictionary, and Bunrui-Goi-Hyo(BGH) is indispensable for these applications. However, such a resource makes it nearly impossible to cover large, and fast-changing linguistic knowledge required for these application systems, depending on text-type and subject domain. Let us take a look at the Japanese thesaurus dictionary called Bunrui-Goi-Hyo(BGH). It consists of 87,743 content words. These words are organized into fine-grained classification, 32,636 semantic classes. The average number of words is thus very few, 2.69 per class. Therefore, considering this resource scarcity problem, semantic tagging of words which do not appear in the resource but appear in corpora has been an interest since the earliest days when a number of large scale corpora have become available.

In this paper, we present a method for semantic classification of Japanese unknown words by combining similarity-based approaches. Here, 'unknown words' are the extracted words which appear in a large collection of Japanese newspapers, but do not appear in the BGH. The method combines three different similarity-based techniques, i.e. (1) edit-distance, (2) *k*-th Reciprocal Nearest Neighbors(RNN), and (3) semi-supervised clustering, to assign each unknown word to a correct semantic class. The outputs obtained by each technique is evaluated by simple voting(Majority). Despite the simplicity of the method, empirical results are very encouraging, and shows that the method can capture more delicate meaning of words.

2 System Description

The method for classifying unknown words into semantic classes consists of three procedures: (1) extracting unknown words from a large collection of Japanese data, (2) collecting systems' outputs which are obtained by similarity-based approaches, and (3) combining the outputs by voting. Figure 1 illustrates a flow of the method.

2.1 Extracting Unknown Words

The first step to classify unknown words into semantic classes is to extract unknown words from a corpus. We choose 15 years Mainichi Japanese Newspapers. All documents were parsed by the syntactic analyzer CaboCha(T.Kudo and Y.Matsumoto, 2003), and all the dependency triples (*rel*,*n*,*v*) are extracted. Here, *n* refers to a noun, and a subject or an object of a verb *v* in a sentence. *Rel* denotes a grammatical relationship between a noun and a verb. We classified *rel* into

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 75–84,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

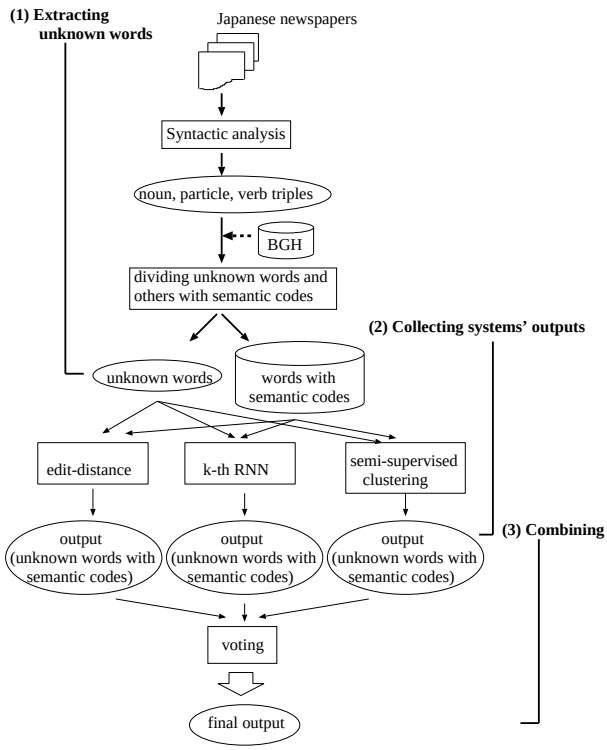


Figure 1: Flow of the method

9 types of Japanese particle, 'ga', 'wo' 'ni' 'he' 'to' 'de' 'yori' 'kara' 'made'. We calculated mutual information value between noun n and verb v using Eq. (1).

$$I(rel,n,v) = \log_2 \frac{\frac{f(rel,n,v)}{N}}{\frac{f(n,rel)}{N}\frac{f(v)}{N}} \quad (1)$$

In Eq. (1), the notation $f(rel,n,v)$ denotes the frequency of dependency triples from a corpus. $f(n,rel)$ refers to the frequency of the noun n followed by rel, and $f(v)$ is the frequency of the verb v. N is the total number of dependency triples. We extracted noun and verb pairs whose mutual information value is larger than a certain threshold. These extracted nouns are candidates of unknown words. We used Bunrui-Goi-Hyou(BGH) as a thesaurus dictionary of Japanese. We regard the extracted noun words which are not classified into any of the BGH classes as unknown words, and used them in the experiments.

2.2 Collecting Systems' Outputs

The second step to classify unknown words is to calculate similarity between an unknown word and each word which is assigned to a semantic class.

1. Edit-distance

Edit-distance is a metric for identifying the orthographic similarity of two words. Typically, one edit-distance corresponds to one substitution, deletion or addition of a character(R.A.Wagner, 1974). It has been widely used to check spelling errors(F.Damerau, 1964). Our edit distance feature represents the edit distance from the unknown word to the closest words in BGH. The unknown word is classified into some semantic class whose distance is smallest among others. Let '香 (fragrance)' be an unknown word, and '香水 (perfume)' and '世界 (world)' be classified into a semantic class A and B, respectively. '香' is translated into '香水' by one addition, '水'. The edit-distance is thus one. On the contrary, the edit-distance of '香' is two, since it is translated into '世界' by one substitution, '香' for '世', and one addition, '界'. '香' is thus judged to be similar to '香水', and assigned to A.

We note that Japanese use four types of writing, *kanji* characters, *hiragana* and *katakana* phonetic scripts, and the Roman alphabet. Of these, kanji characters especially provide semantic clues for unknown word sense. However, Japanese unknown words are relatively short length. Therefore, when one word is compared against others, it is assigned to equal scores. We thus use three types of writing of Japanese except for the Roman alphabet as a default strategy, i.e. when we got ties, we assigned unknown word to the same type of writing.

2. K-th Reciprocal Nearest Neighbors

D.Hindle (1990) presented a method for detecting similar words from text corpora, especially to detect synonyms, called Reciprocal Nearest Neighbors. Our method, k-th Reciprocal Nearest Neighbors(RNN) is based on Hindles' method. Let $n_1^{'}$ and $n_2^{'}$ be a noun. ($n_1^{'}$, $n_2^{'}$) is k-th RNN when $n_1^{'}$ and $n_2^{'}$ satisfy Eq. (2). $sim(n_i^{'},n_j^{'})$ in Eq. (2) is Hindles' noun similarity which is defined by Eq. (3). $sim(rel_i,v_j,n_1,n_2)$ is calculated using mutual information which is shown in Eq. (1). Let $n_1^{'}$ be an unknown word, and $n_2^{'}$ be a word appeared in the BGH. If $n_1^{'}$ and $n_2^{'}$ satisfy Eq. (2), $n_1^{'}$ is classified into a semantic class to which $n_2^{'}$ is assigned. If there are several words $n_2^{'}$ which appeared in the BGH, $n_1^{'}$ is classified into a semantic class whose sim value between $n_1^{'}$ and $n_2^{'}$ is largest among them.

$$(n_1^{'}, n_2^{'}) \in S^k_{n_1^{'}} \cap S^k_{n_2^{'}} \text{ where } S^k_{n_i^{'}} = \{(n_i^{'}, n_j^{'}) \mid n_j^{'} \text{ is topmost } k \text{ according to the descending order of } sim(n_i^{'}, n_j^{'})\} \quad (2)$$

$$sim(n_1, n_2) = \sum_i \sum_j sim(rel_i, v_j, n_1, n_2) \quad (3)$$

$$sim(rel_i, v_j, n_1, n_2) = \begin{cases} min(I(rel_i, n_1, v_j), I(rel_i, n_2, v_j)) & I(rel_i, n_1, v_j) > 0 \text{ and } I(rel_i, n_2, v_j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

3. Semi-Supervised Clustering

We recall that each semantic class of BGH has only a small amount of labeled data, i.e. 2.69 words per class. We thus use semi-supervised clustering, which employs a small amount of labeled data to aid unsupervised learning. There have been a number of work on semi-supervised clustering(S.Basu et al., 2002), (A.Bar-Hillel et al., 2003). We used Basu et al's method, since F.Fukumoto and Y.Suzuki (2006) reported that it is effective for clustering news stories in the task of tracking. The method uses k-means with the EM algorithm.

K-means is a clustering algorithm based on iterative relocation that partitions a dataset into the number of k clusters, locally minimizing the average squared distance between the data points and the cluster centers. The EM algorithm is a method of finding the maximum-likelihood estimate of the parameters of an underlying distribution, from a set of observed data that has incomplete or missing values. The process is assumed to be as follows: First, one Gaussian is chosen out of the K following their prior probability distribution. Then, a data-point(labeled word) is sampled following the distribution of the chosen Gaussian. Let $\chi = \{x_1, \cdots, x_N\}$ be a set of labeled words we want to cluster with each $x_i \in \mathbf{R}^d$. The missing data Z, i.e. unknown word is the cluster assignment of the labeled words. It takes values in $\{1, \cdots, K\}$ and is always conditioned on the labeled word under consideration. The input data for clustering, i.e. each labeled and unlabeled noun word is represented as a vector. Each dimension of a vector is mutual information value of the noun and verb words with dependency triple (rel, n, v) which is calculated in Eq. (1). The k-means objective can be written in Eq. (4).

$$\mathscr{F}_{kmeans} = \sum_{l=1}^{K} \sum_{i=1}^{N} || x_i - \mu_l ||^2 \, p(z_l \mid x_i, \mu_l) \quad (4)$$

$$p(z_l \mid x_i, \Theta) = \begin{cases} 1 & \text{if } l = argmin_h || x_i - \mu_h ||^2 \\ 0 & \text{otherwise} \end{cases}$$

z_l shows l-th missing data, i.e. unknown word. $\{\mu_1, \cdots, \mu_k\}$ represent the k partition centers, and Θ is the parameter set which consists of just the k means $\mu_1, \cdots, \mu_k$.

2.3 Combining the Outputs

The last step to classify unknown words is to combine the outputs. We used voting, as it has demonstrated to be an effective technique for improving accuracy for many NLP applications such as morphosyntactic word class tagging(H.V.Halteren et al., 2001), parsing(J.C.Henderson and E.Brill, 1999) and WSD(T.Pedersen, 2000). The method is quite simple: each of the three approaches votes for the most probable class given the input, and unknown word is classified into a class that receives a majority of the votes.

3 Experiments

3.1 Experiments Set Up

We choose 15 years(from 1991 to 2005) Mainichi Japanese Newspapers which consist of 1,595,817 documents. We used 5 years(from 1991 to 1995) Mainichi Newspapers as a training set to estimate two types of threshold. One is mutual information value, and another is the appropriate number for k of the k-th RNN. The remaining data is a test set to evaluate each method using these thresholds. All documents were parsed by a syntactic analyzer called CaboCha(T.Kudo

and Y.Matsumoto, 2003), and dependency triples are extracted. The total number of dependency triples extracted from a training set and test set is 11,091,187 and 28,234,127, respectively.

We calculated mutual information value between noun and verb words using a dependency triple (rel, n, v), and extracted noun and verb pairs whose mutual information value is larger than a certain threshold. The threshold value we choose was 5×10^{-6}. The threshold is determined as follows: we create a list which consists of verb and noun words, frequencies of each words, and their co-occurrence frequencies using the extracted triples from the training data. We note that frequency of noun, verb, and their co-occurrence do not have a fixed range. We thus apply the following normalization strategy to bring these three values into the range $[0, 1]$.

$$f_{new} = \frac{f_{old} - f_{min}}{f_{max} - f_{min}} \quad (5)$$

We selected topmost different number of verb and noun pairs according to their mutual information value, and the value that optimized precision was chosen for the threshold. The value was 5×10^{-6} . The frequency of noun, verb, and their co-occurrence which satisfies the threshold was, 6, 3, and 3, respectively. We extracted triples whose element satisfies these values. As a result, we obtained 1,464,761 triples. Of these, there are 81,272 different noun words.

For the k-th RNN, we determined the appropriate number for k as follows: we tested different value of k (k = 1 to 30) using a training data, and the value that optimized the global accuracy for identifying semantic class of unknown words was chosen. The value of k was 20.

We used Bunrui-Goi-Hyou(BGH) as a thesaurus dictionary which consists of 87,743 content words with 4 level depth. We used semantic classes of noun words with the deepest level. There are 18,572 classes in all. 40,790 out of 81,272 nouns extracted from the training data are classified into 11,305 classes. The remaining words, 40,482 words are unknown words which are not classified into any of these 18,572 classes. We selected topmost 30,000 words according to their frequency. We manually evaluated these words. The evaluation is made by two authors. The classification is determined to be correct if two author judges agree. As a result, 20,321 words which should be classified into one of the 11,305 classes. We regard 20,321 words as unknown words, and used them in the experiments.

Table 1 shows the number of noun words, the number of classes, and examples of each class. Table 1 illustrates that the maximum number of words per class was 6. In a similar way, there are 68 classes, each of them has more than 5 noun words. 'Examples of each class' refers to an example word which belongs to each class, and 'Sense' shows its sense. The number of each example denotes semantic codes assigned to each class. Each dot of the semantic code refers to a hierarchical level of semantics in the BGH.

3.2 Basic Results

Table 2 summaries classification results. The top of the table shows the average performance when we used labeled words per class is 4. In a similar way, the middle and the bottom is the results when we used labeled words per class is 3 and 2, respectively. 'BGH' refers to the results that we tested unknown words which are created by deleting semantic code from 40,790 words assigned to the BGH. 'Unknown words' shows the results which we used unknown words extracted from Japanese corpora. 'Ratio' refers to the ratio of the number of labeled words and that of unlabeled word per class. For example, 'ratio = 3' shows that the number of unknown word per class is 1, and that of labeled word is 3. '# of classes' is the number of different classes in the input. Each value which is shown in Table 2 denotes the average accuracy: we randomly selected classes. This process is repeated 1,000 times for each ratio and each number of classes. The average accuracy is a ratio that the average number of unknown words which is assigned correctly divided by the total number of unknown words over 1,000 trials.

As can be clearly seen from Table 2, more labeled words improves the overall performance at every number of words per class. When the ratio equals to 3, the average accuracy is higher than .798 and that of 1 is at most .746. The result is statistical significance, i.e. P-value $\geq$ .005 using micro sign test. As we have shown in the examples of Table 1, each meaning of the class is very delicate. Therefore, these results show the effectiveness of the method. Table 2 also shows that the more the number of classes is, the lower the accuracy is. However, more labeled words keeps

Table 1: # of noun words and # of classes

Words per class	# of classes	Examples of each class	Sense
6	6	1.4400.1.1 {‘お宅’, ‘家’, ‘住居’, ‘住’, ‘居’, ‘宅’ }	housing
5	68	1.4400.8.1 {‘王宮’, ‘宮’, ‘宮城’, ‘御所’, ‘皇居’ }	palace
4	428	1.4400.6.1 {‘屋敷’, ‘舘’, ‘邸’, ‘邸宅’ }	mansion
3	1,576	1.4400.9.1 {‘アジト’, ‘隠れ家’, ‘借住い’ }	hideout
2	4,661	1.4400.3.1 {‘アパート’, ‘マンション’ }	apartment
1	4,566	1.4400.1.3 {‘別荘’ }	cottage
Total	11,305	———	

performance. This is because when the ratio is 3, the worst result is .743 and that of 1, the accuracy is .668. Comparing to the performance between ‘BGH’ and ‘Unknown words’, there are no significant difference between them, as the difference is at most .054(the ratio is 1 and the number of classes are 428 and 1,576). This shows that the method is effective for words which do not appear in the BGH.

3.3 The Efficacy of Voting

We used three approaches based on similarity: (1) edit-distance, (2) *k*-th RNN, and (3) semi-supervised clustering. Moreover, we combined the outputs by voting. The contribution of each approach, especially the use of voting is best explained by looking at each result.

Figure 2 plots the result of combining approach when we used ‘Unknown words’ and ‘68 classes’. ‘Combining(labeled:unlabeled=4:1)’ shows that the number of unknown word per class is 1 and that of labeled word is 4. Each dot in Figure 2 refers to the accuracy of the single best approach. Figure 2 shows that for very small labeled words, ‘(labeled:unlabeled=1:1)’, combining method is more beneficial than the result by ‘(labeled:unlabeled=4:1)’, while combining by both data shows better performance than any single approach.

Figures 3, 4, and 5 illustrate the results of ‘unknown words’ against the different number of labeled words per class(Ratio = 1). Overall performance by system combination was better than all the individual approaches, especially the result with ‘unknown words & 428 classes’ (Figure 3) shows that combining method was better than all the individual approaches with choosing any number of classes.

When the number of labeled words per class is 4 (Figure 3), the performance of edit-distance was

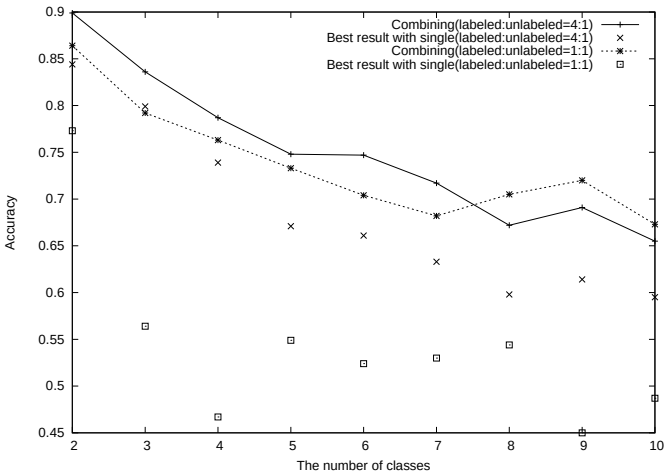


Figure 2: The Efficacy of Voting at different rate of containing labeled words (Unknown words, 68 classes)

better than the other individual approaches with choosing any number of classes. We recall that we used three types of writing of Japanese, kanji characters, hiragana and katakana phonetic scripts as a default strategy, i.e. when we got ties, we assigned unknown word to the same type of writing. This is because Japanese unknown words are relatively short length, and the assignment of unknown words often tie. Figure 6 shows the result of edit-distance using ‘unknown words’ against the different number of labeled words per class (Ratio = 1).

In Figure 6 ‘correct(without tie)’ refers to the ratio without tie against the ratio of unknown words which are classified correctly. In a similar way, ‘incorrect(without tie)’ shows rates without tie against the ratio of unknown words which are not classified correctly. As can be clearly seen from Figure 6, most of the errors using edit-distance are occurred when the assignment of unknown words ties. This indicates that our de-

Table 2: Basic Results

Classification 428 classes	Ratio	# of classes									
		2	3	4	5	6	7	8	9	10	Avg.
BGH	1	.875	.818	.776	.742	.691	.703	.676	.663	.601	.724
	2	.899	.826	.802	.715	.733	.713	.674	.672	.656	.743
	3	.906	.871	.833	.810	.772	.762	.788	.738	.777	.806
	Avg.	.893	.838	.804	.756	.732	.726	.713	.681	.678	——
Unknown words	1	.856	.723	.731	.756	.632	.627	.669	.578	.636	.670
	2	.886	.828	.734	.757	.762	.773	.773	.725	.726	.774
	3	.950	.889	.740	.756	.723	.785	.853	.745	**.743**	**.798**
	Avg.	.897	.813	.735	.756	.706	.728	.765	.683	.702	——

Classification 1,576 classes	Ratio	# of classes									
		2	3	4	5	6	7	8	9	10	Avg.
BGH	1	.869	.851	.776	.796	.713	.689	.674	.681	**.668**	**.746**
	2	.856	.804	.794	.747	.762	.702	.680	.657	.676	.742
	Avg.	.863	.828	.785	.772	.738	.696	.677	.669	.672	——
Unknown words	1	.845	.950	.815	.543	.636	.581	.665	.607	.589	.692
	2	.845	.823	.681	.696	.740	.756	.668	.656	.772	.732
	Avg.	.845	.887	.748	.748	.738	.719	.717	.617	.731	——

Classification 4,661 classes	Ratio	# of classes									
		2	3	4	5	6	7	8	9	10	Avg.
BGH	1	.856	.723	.731	.756	.632	.627	.669	.608	.636	.693
Unknown words	1	.852	.836	.679	.672	.689	.609	.629	.626	.591	.687

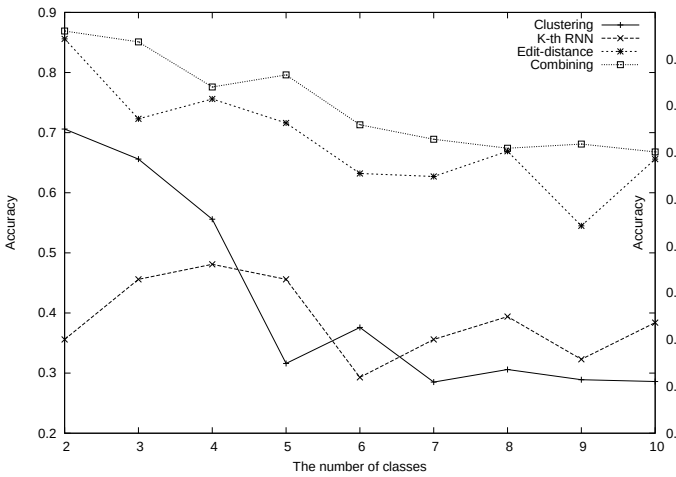


Figure 3: The Efficacy of Voting (Unknown words, 428 classes)

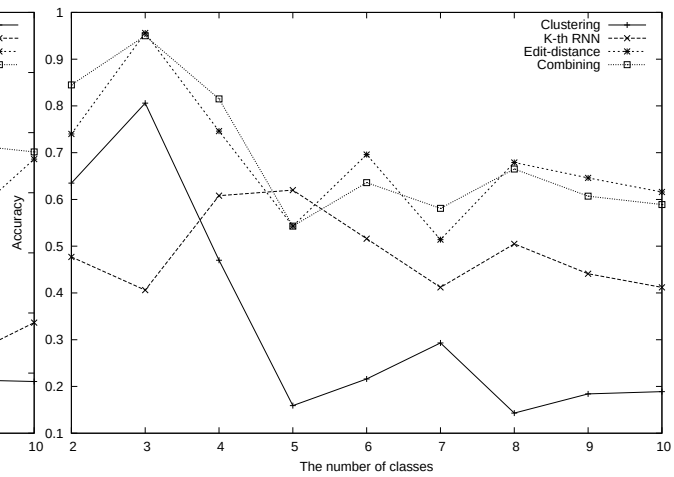


Figure 4: The Efficacy of Voting (Unknown words, 1,576 classes)

fault strategy is not so effective(at most 9.7%, 4 classes), while the edit-distance itself is very effective among other individual approaches.

The performance of *k*-th RNN was lowest among all the individual approaches, while semi-supervised clustering is more sensitive to the number of classes than edit-distance and *k*-th RNN. One reason behind this is that we restricted the number for *k* to 20 because of run time efficiency. As a result, for example, when the number of classes is 2 in Figure 3, we could not find simi-

Table 3: Examples obtained by *k*-th RNN

k	Unknown word	Similar word
1	ロイター通信 (Reuters)	通信社 (news agency)
2	銅メダル (bronze medal)	メダル (medal)
3	ヘリ (helicopter)	ヘリコプター (helicopter)
4	悪影響 (negative effect)	害 (ill effect)
5	憶測 (guess)	うわさ (rumor)
6	コメ (rice)	米 (rice)
7	昨年度 (last year)	昨年 (last year)
8	ゴミ (garbage)	空き缶 (empty can)
9	日々 (daily)	毎日 (every day)
10	保留 (suspension)	非公開 (nondisclosure)
11	府警 (Osaka police)	県警 (prefectural police)
12	イラン (Iran)	パキスタン (Pakistan)
13	ハワイ (Hawaii)	シンガポール (Singapore)
14	思惑 (thinking)	予想 (expectation)
15	ワシントン (Washington)	北京 (Beijing)
16	今季 (this season)	今年 (this year)
17	新進党 (NFP)	党 (party)
18	アフガニスタン (Afghanistan)	アフガン (Afghan)
19	民社党 (DSP)	党 (party)
20	同社 (company)	会社 (company)

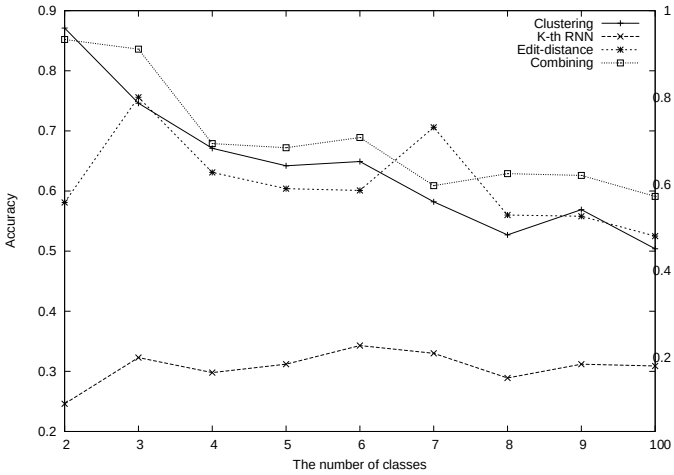


Figure 5: The Efficacy of Voting (Unknown words, 4,661 classes)

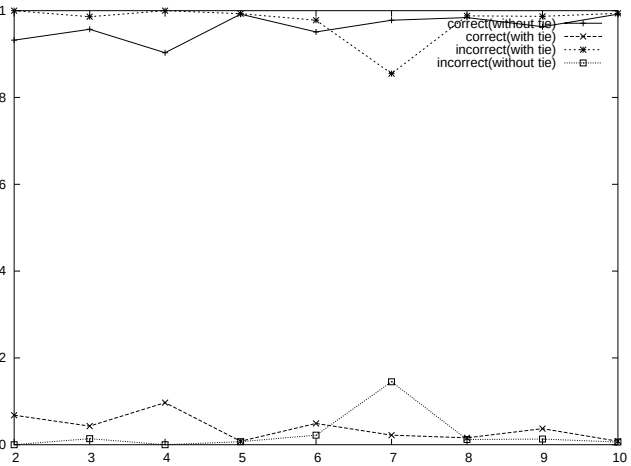


Figure 6: The Efficacy of edit-distance (Unknown words, 428 classes)

lar words of 1,217 out of 2,000[1], and the accuracy of *k*-th RNN was .356. However, the *k*-th RNN is effective when words exist within the *k*-th similar nouns. Table 3 illustrates examples of the result obtained by *k*-th RNN.

'k' in Table 3 refers to the number for *k* in the *k*-th RNN, and 'Similar word' shows the output extracted by the *k*-th RNN, i.e. a word within *k*-th similar nouns. This result claims that the accuracy of *k*-th RNN depends on the number of *k*. Varying the number of *k* for further efficacy gains is definitely worth trying with our approach.

3.4 The Efficacy of Clustering

Our hypothesis using semi-supervised clustering is that it might be able to compensate for drawback of resource scarcity, i.e. a small amount of

[1]We repeated 1,000 trials for 2 unknown words. Thus, the total number of unknown words is 2,000.

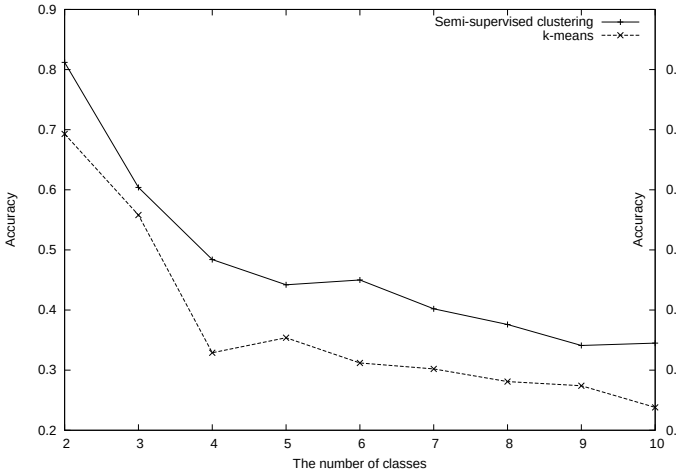


Figure 7: Semi-supervised clustering v.s. *k*-means (Unknown words, 68 classes)

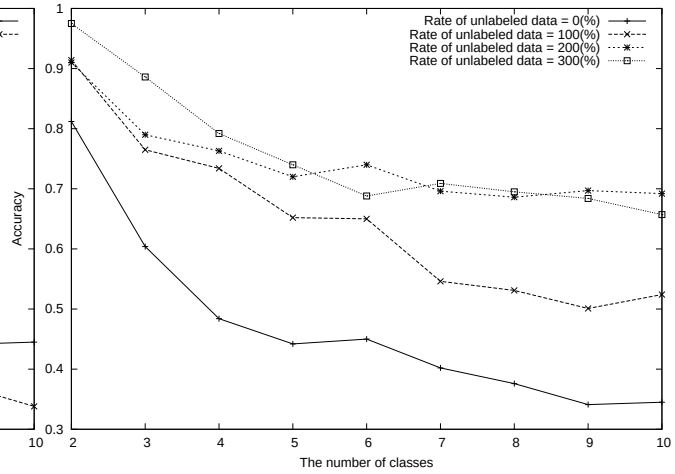


Figure 8: The Efficacy of Clustering (Unknown words, 68 classes)

labeled words per class. To examine the effect of semi-supervised clustering, especially the effect of labeled words to aid and bias the clustering of unlabeled words, we compared the results by our method with *k*-means which is shown in Figure 7. We randomly selected classes according to the number of classes. 'Accuracy' in Figure 7 refers to the average correct unknown words over 1,000 trials. We set *k* of the *k*-means to the number of classes. For both methods, the proportion of unlabeled to labeled words is 1 to 1.

As can be clearly seen from Figure 7, the result with semi-supervised clustering was consistently better that that with *k*-means, the maximum accuracy difference between them was .155 (# of classes = 4), and that of minimum was .046 (# of classes = 3).

Finally, we evaluate the effect of semi-supervised clustering at different rate of containing unlabeled words('Unknown words' and '68 classes'). For each class, we created test data, i.e. unlabeled words extracted from the BGH by deleting semantic code. We used test data, together with labeled and a certain amount of unlabeled words, and examined whether the test data(words) are correctly classified using by both labeled and a certain amount of unlabeled words. Figures 8 shows the results. 'Rate of unlabeled data' shows a rate that the number of unlabeled words against the number of test words. For example, '0%' shows that test words are classified by using only labeled words. The accuracy in Figure 8 refers to the average accuracy over 1,000 trials.

Figure 8 shows that more unlabeled words helps overall performance of the clustering at any number of labeled words per class. However, when the ratio of unlabeled data are 200% and 300%, there is no statistical significance between two plots, as the result was P-value $\geq$ .005 using micro sign test. This indicates that the number of unlabeled words is three times higher than that of test words, little gains by semi-supervised clustering, and indeed on the largest number of unlabeled words, clustering actually hurts accuracy. We used mutual information value of the noun and verb words as each dimension of a vector which is the input in the clustering. However, mutual information is not the only possible weighting approach. Investigating other criteria, such as χ^2 method or distributional similarities(D.Lin, 1998),(D.Baker and A.McCallum, 1998) may improve performance.

4 Related Work

With the exponential growth of readily available on-line text, research activities for classifying unknown words into semantic classes by using corpus-based statistics are very popular. One type of semantic classification of unknown words uses distributional similarity(D.Hindle, 1990), (D.Lin, 1998), (D.Ido et al., 1999), (K.J.Chen and C.J.Chen, 2000). Similarity measures based on distributional hypothesis compare a pair of weighted feature vectors that characterize two words. Features typically correspond to other words that co-occur with the characterized word in the same context. It is then assumed that different words that occur within similar contexts are semantically similar. Lin proposed a word similar-

ity measure based on the distributional pattern of words which allows to construct a thesaurus using a parsed corpus. Moreover, he compared the result of automatically created thesaurus with WordNet and Roget, and reported that the result is significantly closer to WordNet than Roget Thesaurus is. However, as M.Geffet and I.Dagan (2004) pointed out, distributional similarity captures a somewhat loose notion of semantic similarity. For example, words of similar semantic types, such as *company* and *government*, tend to come up as distributionally similar, even though they are not substitutable in a meaning preserving sense. To capture more delicate meaning, the method presented in this paper uses three approaches based on similarity measure, each of them is different in the aspect of unknown words.

Another type of previous work on classification of unknown words uses machine learning(ML) techniques, which automatically learn semantic information from on-line text corpora. S.Baluja et al. (1999) et. al proposed a method to identify proper names in text using a decision tree classifier. They incorporate three types of features: word level, dictionary-level, and POS information. Similar to S.Baluja et al. (1999)'s approach, J.Toole (2000) used a decision tree to classify unknown words into two categories, misspellings and names using decision trees. They reported that the result using data from live closed captions which replete with a wide variety of unknown words showed the effectiveness of the method. S.Stevenson et al. (1999) investigated various ML techniques to automatically classify a set of lexical semantic verbs, based on distributional features extracted from a corpus.

For the systems based on ML, the quantity and quality of training corpora, i.e. tagged data, affect their performance. While the amount of available on-line text has been increasing at a dramatic rate, the size of training corpora typically used for learning has not. This is because it needs the potentially large cost of annotating data for those learning methods that rely on labeled text. The method presented in this paper does not require a number of training data, but instead uses a small amount of labeled data together with unlabeled data.

Previously studied techniques for combining the outputs include (i) voting techniques(H.V.Halteren et al., 2001), (ii) switching among several systems according to confidence values they provide(J.C.Henderson and E.Brill, 1999), and (iii) stacking techniques(H.Wolpert, 1999), (T.Utsuro et al., 2002) which train a second stage classifier for combining outputs of classifiers at the first stage. The process of system combination consists of two steps: collection of different systems, and combination of the outputs obtained by several systems. The first step is especially important for overall performance, and needs to collect systems which behave as differently as possible. Such methods are promising, although to our knowledge they have not been applied to the resource scarcity problem of the thesaurus dictionary.

5 Conclusion

We proposed a method for semantic classification of unknown words which are not appeared in the dictionary BGH by combining three approaches based on words similarities. Overall performance by system combination was better than all the individual approaches, especially the result that we used labeled words per class is 4 and unknown words extracted from Japanese corpora('unknown words & 428 classes') shows that combining method was better than all the individual approaches with choosing any number of classes. Moreover, we have found that the performance of edit-distance was highest among all the individual approaches, and semi-supervised clustering was more sensitive to the number of classes than edit-distance and k-th RNN. Future work includes (i) exploring each approach in order to improve the efficacy even for a large number of input classes, (ii) applying the method to other thesaurus such as WordNet to evaluate the robustness of the method, and (iii) applying the results to some applications such as relevant document extraction.

References

A.Bar-Hillel, T.Hertz, N.Shental, and D.Weinshall. 2003. Learning distance functions using equivalence relations. In *Proc. of the 20th International Conference on Machine Learning*, pages 11–18.

B.Regina and K.McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proc. of 39th Annual Meeting of the Association for Computational Linguistics and 10th Conference*

of the European Chaper of the Association for Computational Linguistics, pages 50–57.

D.Baker and A.McCallum. 1998. Distributional clustering for text classification. In *Proc. of 21st ACM International Conference on Research and Development in Information Retrieval(SIGIR'98)*, pages 96–103.

D.Hindle. 1990. Noun classification from predicate-argument structures. In *Proc. of 28th Annual Meeting of the Association for Computational Linguistics*, pages 268–275.

D.Ido, L.Lee, and F.Pereira. 1999. Similarity-based models of cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69.

D.Lin. 1998. Automatic retrieval and clustering of similar words. In *Proc. of 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 768–773.

D.Lin. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 28(3):343–360.

F.Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7:171–176.

F.Fukumoto and Y.Suzuki. 2006. Using comparable corpora and semi-supervised clustering for topic tracking. In *Proc. of 21th International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 231–238.

H.V.Halteren, J.Zavrel, and W.Daelemans. 2001. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, 27(2):199–229.

H.Wolpert. 1999. Stacked generalization. *Neural Networks*, 5:491–497.

J.C.Henderson and E.Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proc. of 1999 Empirical Method in Natural Language Processing and Very Large Corpora*, pages 34–41.

J.Toole. 2000. Categorizing unknown words: Using decision trees to identify names and misspellings. In *Proc. of 6th Applied Natural Language Processing Conference*, pages 173–179.

K.J.Chen and C.J.Chen. 2000. Automatic semantic classification for chinese unknown compound nouns. In *Proc. of 38th Annual Meeting of the Association for Computational Linguistics*, pages 125–130.

M.Geffet and I.Dagan. 2004. Feature vector quality and distributional similarity. In *Proc. of the 20th International Conference on Computational Linguistics*, pages 247–253.

R.A.Wagner. 1974. The string-to-string correction problem. *ACM*, 21(1):165–173.

S.Baluja, V.Mittal, and R.Sukthankar. 1999. Applying machine learning for high performance named-entity extraction. In *Proc. of the Pacific Association for Computational Linguistics*, pages 365–378.

S.Basu, A.Banerjee, and R.Mooney. 2002. Semi-supervised clustering by seeding. In *Proc. of the 19th International Conference on Machine Learning*, pages 19–26.

S.Ikehara, S.Shirai, A.Yokoo, F.Bond, and Y.Omi. 1994. Automatic aquisition of semantic attributes for user defined words in japanese to english machine translation. In *Proc. of 4th Conference on Applied Natural Language Processing*, pages 184–185.

S.Stevenson, P.Merlo, N.Kariaeva, and K.Whitehouse. 1999. Supervised learning of lexical semantic verb classes using frequency distributions. In *Proc. of the SIGLEX-99 Workshop: Standardizing Lexical Resources*, pages 15–21.

T.Kudo and Y.Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proc. of 41th Annual Meeting of the Association for Computational Linguistics*, pages 24–31.

T.Pedersen. 2000. A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. In *Proc of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 63–69.

T.Utsuro, M.Sassano, and K.Uchimoto. 2002. Combining outputs of multiple japanese named entity chunkers by stacking. In *Proc. of 2002 Empirical Method in Natural Language Processing*, pages 281–288.

Using Contexts of One Trillion Words for WSD

Tobias Hawker
School of Information Technologies
University of Sydney
NSW 2006, Australia
toby@it.usyd.edu.au

Abstract

This paper describes how features derived from one trillion words of text can be used in a practical supervised Word Sense Disambiguation system.

Our system is based on substituting into the context of the target word, others ontologically related to it. Words related to the correct sense of the word are likely to occur with its context more frequently in the corpus than relatives of other senses. The Pointwise Mutual Information arising from these frequencies is used to form features for supervised learning. The system retains many of the advantages of supervised systems, while using the Web1T corpus to ameliorate sparse data problems.

Using the Web1T data presents significant processing challenges, however, to which we describe practical solutions. We demonstrate that the features we describe improve the performance of systems when combined with existing local context features.

1 Introduction

While it is clear that more annotated data would improve the performance of Word Sense Disambiguation (WSD) systems, obtaining such data is time-consuming and costly (Chklovski and Mihalcea, 2002). WSD is a task on which supervised methods substantially outperform unsupervised systems. It is likely that even better performance could be achieved were it not for this "data-acquisition bottleneck".

Methods for WSD which can make use of information beyond that available from hand-tagged corpora are therefore highly desirable. In this paper, we introduce a method for acquiring useful features for WSD from a massive quantity of unannotated data, whilst still operating within a supervised framework.

This method takes advantage of a recently released dataset (Brants and Franz, 2006) — Web1T — that is several orders of magnitude larger than those used for previous attempts involving unannotated corpora for WSD: it contains n-gram frequencies derived from one trillion (10^{12}) words.

The Web1T data is used by relying on the insight that each ambiguous word has several semantically related words that may, depending on the context and intended meaning, function as a substitute. The suitability of a candidate for substitution in a particular case can be assessed by replacing the ambiguous word with that candidate in its local context, and determining the plausibility of the resulting word sequence by examining its frequency in a large sample of text. If the resulting sequence is substantially more common than we should expect by chance, it can be inferred that the intended meaning of the original word is the sense it shares with the substitute. Conversely, if the resulting sequence is rarer than we expect, the belief that this is the intended sense can be reduced.

It has been shown (Kaplan, 1955; Choueka and Lusignan, 1985) that it is plausible for humans to perform at least coarse disambiguation tasks reliably using only two words of context on either side of the target. While native speakers must use their knowledge of the world to disambiguate with so little context, such world knowledge cannot be accessed directly. Instead, we attempt to infer it from the patterns of usage in the very large Web1T corpus. Additionally, the system described can make use of up to four words of context in either direc-

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 85–93,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

tion from the target word by using a sliding window.

2 Related Work

This system is not the first to attempt using untagged data for supervised WSD. Approaches such as bootstrapping (Mihalcea, 2002), and distributional (Dagan et al., 1997) or semantic similarity (Patwardhan et al., 2003) have all been investigated, as have the following more closely related methods.

2.1 Bilingual Corpora

Some approaches attempt to determine the sense of ambiguous words by taking advantage of the contextual modeling in machine translation. These may use either parallel corpora (Chan and Ng, 2005) or automatic translation systems (Wang and Martinez, 2006), relying on similar contextual cues as our method, albeit indirectly. This paper applies a much larger, unannotated monolingual corpus, without requiring any translation, human or machine.

2.2 Monosemous Relatives

The system described by Leacock et al. (1998) and others using similar principles (Mihalcea and Moldovan, 1999) have shown that related words may be used to gather additional training examples for WSD from unannotated data. They are however limited to unambiguous substitutes, or those whose usage is strongly dominated by the required sense. These 'monosemous relatives' have also been used in unsupervised systems (Agirre and Martinez, 2004).

2.3 Relatives in Context

Martinez et al. (2006) have recently described an unsupervised method similar to the one presented here, which uses the frequencies reported by queries to web search engines to determine the sense of an ambiguous word. The system is currently confined to nouns. A best-match strategy is used to assign the sense ('best' defined via heuristics as most closely related, longest and most frequent), rather than evaluating the relative frequencies of several substitutes. This approach is not yet able to use gradients of substitutability within a broad set of substitutes.

Unfortunately, there are disadvantages in terms of the reliability, reproducibility and even scalability of systems reliant upon frequencies reported by queries to web-based search engines. The details of the mechanism by which the counts reported by the search engine are derived are not publically available. Search engines may place a restriction on the number of queries that may be made in a given time interval by automatic systems. The numbers retrieved could be subject to variation due to the handling of words in the local context, and the results may change as the search engine index is updated or propagated throughout the distributed systems used to serve web queries. Over time, there may even be algorithmic changes in the search engine used, impacting on the reproducibility of results obtained using web queries.

None of these drawbacks apply to a single released corpus of word sequences and their frequencies, such as the Web1T corpus. This corpus also includes explicit descriptions of such information as frequency cut-offs, methodological decisions, and the total quantity of text used to create the data.

3 Contextual Substitution

As an example to demonstrate the basic mechanism underlying the system described in this paper, consider the sentence fragments around the verb *ruled* in:

the court ruled it was clear that

and

a republic ruled by the people

Two possible synonyms, pertaining to different senses for the verb *ruled*, are *found* and *governed*. It would be expected that in a sufficiently large quantity of text, the fragments:

the court found it was clear that

and

a republic governed by the people

would be substantially more common than the sequences:

the court governed it was clear that

or

a republic found by the people

We can thus use the frequencies relating to these substitute expressions as information for a WSD system - where the task is to discriminate between senses.

In a way, implicit language models are being evaluated for each instance of a word needing disambiguation: the sense is chosen whose substitutes are associated with the most likely models.

Fortunately, the substitute word does not have

to be unambiguous. Since the meaning of the entire sequence is likely to be well determined, it is unlikely that a wide variety of senses will occur frequently in the position occupied by the target word. Essentially, the context disambiguates the word, and only allows words with a limited range of meanings to be substituted. Church et al. (1994) show that Pointwise Mutual Information (PMI) is a suitable measure to capture the degree to which a given word may substitute for another; we thus adopt PMI as the quantitive basis for the system.

The supervised system we present in this paper involves the following steps. These steps are explained in more detail below.

1. Choose a set of related words for each ambiguous lemma to be disambiguated,
2. Extract the local context around each ambiguous word's instance,
3. Find the frequency of the context where the ambiguous word has been replaced by each substitute in the large sample of text, and the context frequency irrespective of the target word,
4. Form features relating to the degree of substitutability implied by these counts,
5. Train a statistical model for predicting the class of new examples based on these features.

This approach differs in a number of ways from past methods for supervised WSD. There is no attempt to perform POS tagging or use any other linguistically sophisticated processing. These tasks are arduous on the very large corpus and not necessarily helpful. Meaningful parsing for example is virtually impossible with only 5-token sequences. This effectively means that the only annotation required from the training data is the sense tag.

This method also differs from many previous approaches to using Web-scale Corpora in that we are interested in the differing degree of substitutability of all potential synonyms in the actual context, and not in trying to choose a particular substitute as 'best' (Martinez et al., 2006), nor to find additional training examples (Mihalcea and Moldovan, 1999).

3.1 Related Words

In the experiments described in section 5, related words were obtained from WordNet (Fellbaum, 1998), though as the nature of the WordNet relationship is not explicitly encoded in the system, any source of potential synonyms could be used. The minimum level of relatedness used in experiments was determined by practical considerations: as the number of features is quadratically related to the number of substitutes (see section 3.3.1) we chose criteria that yielded tractable vocabularies for each part of speech. For verbs, this included only words that WordNet listed as synonyms. For nouns, we used all synonyms and immediate hypernyms and hyponyms. For adjectives, synonyms and 'similar' words were used.

Ontologically related words that are not synonyms will not necessarily share distributions particularly similar to the target word. This is more of a problem for systems which try to use them as additional instances to train from, as irrelevant examples may be introduced into the system (Mihalcea, 2004). Mihalcea and Moldovan (1999) refer to preliminary experiments with various ontological relationships, but choose to use synonymy relations only in the final system. However, in our system, surrogates which are consistently poor substitutes for the target word are unlikely to occur in patterns which introduce very much noise. Moreover, machine learning algorithms are frequently more robust to noisy features than noisy training examples.

3.2 Local Context for Substitutability

We used sliding windows to find each n-gram that includes the target word in both training and test data. As the n-gram data available consists of 5 or fewer word sequences, the maximum amount of context required is 4 words on either side of the target.

3.3 Frequency Determination

Our system constructs and retrieves counts for all possible n-grams where the word to be disambiguated is replaced by each of the substitue vocabulary items. Simple morphological analysis is performed on the words being used, so that inflections for number or tense are consistent with the target for all candidate substitutes.

The quantity of data used to describe the catalog of n-grams and their respective frequencies in the Web1T data is considerable: even in gzip-compressed form, the dataset is approximately 25GB. The n-grams and their corresponding frequencies are stored in separate files, each contain-

ing the text and frequencies of 10 million unique sequences. Disk access and search times mean that scanning the data in its original form for every n-gram sequence as it is required is completely impractical.

Finding the PMI between the context and the substitute requires the determination of how often the context occurs with *any* word in the position occupied by the target word. The system is thus required to retrieve both completely specified and partially specified (that is, *wild-card*) patterns from the Web1T corpus. This is achieved by accumulating the patterns required before any examination of the data takes place. The solutions to these substantial problems of scale are discussed further in section 4.1.

3.3.1 Feature Construction

Features derived from the Web1T corpus are all based upon the PMI of each candidate substitute with the particular context of the ambiguous target under consideration. For a given substitute and context, an overall PMI is determined as a single quantity, obtained by simply adding the PMI together from each window of each size covered in the data:

$$\begin{aligned} PMI &= \sum_{n=2}^{5}\sum_{i=1}^{n}\log_2\frac{\text{observation}_{n,i}}{\text{expectation}_{n,i}} \\ &= \sum_{n=2}^{5}\sum_{i=1}^{n}\log_2\frac{\#(\text{sub}+\text{context}_{n,i})}{p(\text{sub})p(\text{context}_{n,i})N_n} \end{aligned}$$

Here n represents the window size (varying from 2 to 5), i is the position within the window, and N_n indicates the total number of n-grams present in the corpus for a given value of n. Following Church et al. (1994) the Maximum Likelihood Estimate (MLE) is used for both probabilities in the denominator. $p(\text{sub})$ is estimated from the unigram frequency of the substitute word, while $p(\text{context})$ is obtained from the underspecified, wild-card counts. Taking one of the examples used above, "the court ruled it was clear that", and the substitute "found", the possible trigram fragments involving the target word are "the court found", "court found it" and "found it was". The frequencies of these exact trigrams constitute $\#(\text{sub}+\text{context}_{3,\{1,2,3\}})$. $p(\text{sub})$ is the unigram frequency of "found" divided by the total unigram count for the corpus. The counts for the wildcard context fragments "the court *", "court * it" and "* it that" are transformed into the context probabilities $p(\text{context}_{3,i})$ (where i is 1, 2 and 3 respectively) by dividing by the total number of trigrams (N_3).

Features were also created that harnessed the idea that it is not only the level of substitutability for each candidate word that is useful, but also that it may be informative to recognise that some words are better substitutes than others. This information was captured by adding additional features consisting of the pairwise differences between PMI values for all candidate substitute words. To further draw the differing levels of substitutability into relief, features representing the rank of each pair's PMI difference were also included.

To enumerate all possible relationships for these comparative feature types, the number of comparisons that must be made for x substitute words is $\frac{x^2-x}{2}$, and twice this number of features are introduced. This quadratic feature set size serves as a limiting factor on the maximum size of the substitute vocabularies for each word.

The real valued features created by the PMI itself, the pairwise differences, or the rankings of those differences can be used directly in various machine learning algorithms. However, particularly when the number of data points is low, it can be difficult for a learning algorithm to determine the influence of different values. Each real-valued feature is thus discretised into simple binary features for machine learning; the process is further described in section 4.2.

4 Implementation

The system developed to extract relevant n-gram frequencies from the Web1T corpus and derive features from them efficiently handles a huge quantity of data. The important characteristics of this approach are described in the following sections.

We will make the software for both efficiently creating, and then performing these queries available[1] to the community.

4.1 Efficiently Querying 1 Trillion Words

The Web1T data includes, for all n-gram window sizes, a directory of compressed text files, each containing n-grams and their frequency counts for 10 million instances. Depending on the size of the

[1] `http://www.it.usyd.edu.au/~toby`

window, this can mean up to 132 files, or more than 1.3 billion n-grams. Although an index of the starting n-gram in each file is included in the distribution, the cost in computer time of scanning through these text files directly, the speed with which data can be read from hard disk, and the CPU burden of decompression makes random access to the Web1T data prohibitive for any more than a small number of queries.

Moreover, the Web1T data is stored alphabetically in a case-sensitive fashion; this means that sequences differing only in capitalisation may be represented by separate entries in completely different locations in the data.

These constraints mean that any practical system cannot perform a single query at a time. We must thus attempt to make a single pass through the corpus. The number of queries for which counts must be retrieved in this single pass can also very easily be unwieldy. This means that compiled regular expressions or other finite-state approaches are rendered impractical. This is particularly unfortunate, as regular expressions are well suited to the wild-card queries we require when estimating the context probability in the PMI calculations.

Our solution is to generate all the string queries in advance from both training and test data, including both complete n-grams and the relevant wild-card queries. When extracting counts from the Web1T corpus, the fully-specified and wild-card queries are stored in separate hashed data structures. A single pass is made over the data, decompressing each file into memory only once. Each n-gram sequence is checked against the exact and wild-card queries, and if a match is found the relevant totals are updated. Using this system, we extracted the counts for more than 20 million queries from the 5-gram data in around three hours on a single GNU/Linux server.

4.2 Discretisation

The features derived from the Web1T corpus are universally real-valued – even though the ranks are integers they are ordered, while the PMI and associated pairwise differences are continuous values. To ensure the machine learner can make maximum use of the features, we thus apply entropy-based multi-interval discretisation (Fayyad and Irani, 1993).

This discretisation process employs the *Minimum Description* or *Minimum Message Length* principle to approximate a quantitative version of Occam's razor. The scheme finds partition points where the (theoretical) message length required to describe the partitions and encode the class label data given this division is shorter than that of encoding the data in the absence of any information about the partitions. The messages and encodings do not have to be calculated explicitly; instead we find the minimum length of a message that could be constructed using an optimal encoding.

Each feature being discretised is considered independently. Points along the scale are considered as candidate cut points, to divide the instances into two groups either side of it. For each candidate point, the message length (or entropy) reduction from describing the two groups separately is compared to the increase required to describe the partition cuts. If the cut yields groups that are more uniformly distributed after the partitioning than before, it is possible that the overall message length may be reduced.

We select the partition that yields the best reduction in message length, and then recursively consider the possible partitions of each resultant group. The algorithm terminates when there are no further cuts which result in an improvement in message length. If no suitable cut points are found at the top level, the feature is discarded, and not used in classification. This process thus ensures that candidate words which are particularly poor at discriminating between senses of the target word are not included in the statistical models constructed.

We have implemented efficient software to perform this discretisation as a standalone package, which we are making available to the community.

5 Experiments

5.1 WSD Evaluation

Two sets of experiments were performed to evaluate the efficacy of a system using the features described above. Both experiment sets are from SENSEVAL-3 – the English *Lexical Sample* (Mihalcea et al., 2004) and *All Words* (Snyder and Palmer, 2004) tasks. To gain insight into the utility of the Web1T corpus approach, experiments were also performed with a system that used the local context surrounding the lemma as features directly; again with a window of four words on either side of the target. The features for this sys-

Task	Test Instances	Baseline	Web1T	Local	Joint	Joint Δ
Lexical Sample nouns	1807	0.542	0.674	0.652	**0.679**	+0.137
Lexical Sample verbs	1978	0.565	0.659	0.699	**0.706**	+0.141
Lexical Sample adjectives	159	0.497	0.516	**0.572**	0.541	+0.044
Lexical Sample all	3944	0.552	0.660	0.672	**0.687**	+0.135
1st Senseval-3 system	*3944*	*0.552*	*N/A*	*N/A*	*72.9*	*+0.177*
2nd Senseval-3 system	*3944*	*0.552*	*N/A*	*N/A*	*72.6*	*+0.174*
All Words	2041	0.624	**0.641**	0.640	**0.641**	+0.017
1st Senseval-3 system	*2041*	*0.624*	*N/A*	*N/A*	*0.652*	*+0.030*
2nd Senseval-3 system	*2041*	*0.624*	*N/A*	*N/A*	*0.646*	*+0.022*

Table 1: WSD Performance. The accuracy for comparison systems is shown in the "Joint" column

tem included combinations of the context words, their location relative to the target and their (automatically assigned) part-of-speech tags. Finally, experiments were performed that combined these two systems, by including both groups of features in the classifier models.

The lexical sample task consists of 57 lemmas: 32 verbs, 20 nouns and 5 adjectives. Training data is supplied for the same set using the same sense inventory, and the system was trained using this data.

The experiments for the All Words task used the SemCor corpus (Miller et al., 1993) as training data. To yield results for all words, models were trained for a subset of polysemous lemmas: those that have more than 100 annotated instances in SemCor. To reduce unnecessary computation, we did not train models for those lemmas in this category which did not appear in the test data. This results in 93 models: 48 nouns and 45 verbs. Lemmas in the test data for which no model was trained are all assigned senses using the WordNet first-sense heuristic.

5.2 Machine Learning

All experiments were performed using the `MegaM` Maximum Entropy Model Optimization package (Daumé III, 2004) for classification. No feature selection was performed beyond the entropy-based supervised discretisation of real-valued features.

5.3 Feature Construction

The Web1T-based system uses only three types of features, each derived from the PMI inferred for the substitutability of the substitute word in the current context from the n-gram data. Although all feature types are initally real-valued, they are discretised according to their distribution in the training data.

The first feature type consists of a direct representation of the PMI quantity itself. As the logarithm base is fixed at 2, this quantity is a measure in bits. It can be negative if the substitute word is seen in the context less than is expected by chance. The next feature is the comparison of each substitute with every other substitute. This is a simple subtraction between features of the first type, and so again is a quantity of bits, this time for each pair of lemmas. The final feature type, the ranking of the pairwise comparisons, is dimensionless, but serves to indicate which pairs of lemmas are most divergent in their degree of substitutability, independent of the quantities involved.

6 Results and Discussion

Results of our systems on the SENSEVAL-3 English Word Sense Disambiguation tasks are given in Table 1. Results on a sample of individual lemmas from the Lexical Sample task for the Web1T system are given in Table 2.

6.1 Lexical Sample

The overall accuracy of the Web1T-based system on the Lexical Sample task is 0.660 for fine-grained evaluation. Recall, precision and accuracy are all identical for these results as our system gives an answer for every test example. In comparison to the supervised systems participating in the lexical sample task at SENSEVAL-3, the score is reasonably good for a single, local feature type, yielding results that are approximately in the middle with respect to results obtained by systems participating in the Lexical Sample task.

A majority of words had improvements over the baseline for the majority of lemmas in each of these parts of speech, with a smaller number of lemmas where performance either stayed level or declined. Aside from the different rules for vocabulary extension discussed above, the system han-

dles lemmas from different parts of speech identically. There is thus no in-principle difficulty with extending the system to also disambiguate adverbs, provided sufficient training data is available.

To investigate the performance further, the variance in performance for different words was compared to the factors shown in the columns of Table 2 (Table 2 shows only a sample of lemmas). We hypothesised that the system's performance should be dependent on the number of training examples available for building the model used to classify each lemma, as the number of features is somewhat larger than the number of training examples. We compared the improvement with respect to the baseline against the number of training examples used for each lemma. However, there is only a weak correlation ($R = 0.07$) between these two quantities, which is not statistically significant.

Similarly, the correlation between improvement against the baseline and the number of vocabulary items available for substitution (with which the number of features also monotonically increases) is even weaker: $R = 0.001$, and is certainly not statistically significant. This is somewhat surprising, as it indicates that there may be diminishing returns in accuracy as the number of potential synonyms is increased. It is possible that there are only a reasonably limited set of words for each lemma which will substitute preferentially enough for one or more senses and thus allow for improved ambiguity resolution.

The negative correlation of improvement against the baseline and the number of classes is not significant but is somewhat stronger ($R = -0.18$). The only statistically significant correlation we measured against improvement beyond the baseline is that with the baseline itself: $R = -0.44$, $p < 0.05$, which is not entirely unexpected, as for words with higher baselines, the same level of error reduction corresponds to a smaller accuracy improvement.

The local context system outperformed the Web1T-based system, indicating that high-precision specific knowledge of the words surrounding an ambiguous word can be more useful than information derived from even a truly massive corpus. This is not consistent across all parts of speech however; nouns are disambiguated more accurately by using the corpus than the concrete local features.

The best performance in our experiments however, is obtained when the two approaches are combined. This yields a result of 0.687 over all parts of speech. The information in both approaches comes from only a few words of local context, yet it seems that evaluating the abstract substitutability in the context can substantially assist when already resolving word sense by directly examining aspects of the same context. This combination system outperforms most of the systems which participated in the Lexical Sample task at SENSEVAL-3. The results of the top two systems participating in each of these tasks at SENSEVAL-3 is also given for comparison.

6.2 All Words

As shown in Table 1, the baseline reported for the All Words task at SENSEVAL-3 was 0.624. The Web1T-based system's accuracy on the same test set was 0.641, an improvement of 0.017. Combination with local context features on this task did not lead to a performance improvement.

Only two of the systems participating in SENSEVAL-3 achieved higher scores on the All Words task, and indeed, most did not outperform the baseline (Snyder and Palmer, 2004). This result is therefore encouraging, particularly when as the Web1T-based system only involves features from within a sliding window of four words from the target. This is particularly pertinent given that the All Words task suffers from a somewhat narrow set of domains (only 3 documents are used as test data) and thus approaches that can use broader contextual information to make inferences about the domain should have an inherent advantage.

As in the lexical sample task, many lemmas for which the system made context-based predictions either improved their performance or remained steady compared to the baseline, while a few did decline in accuracy.

7 Conclusions and Future Work

The system we have described in this paper provides a new mechanism for using massive quantities of non-sparse information about the sense implied by the local context of an ambiguous word. As we have shown, the features derived from this information are suitable for use in supervised WSD systems, even in the absence of any traditional features. Combining this system with an-

Lemma	Training Examples	Classes	Substitutes	Baseline	Accuracy	Improvement
paper.n	232	7	85	0.256	**0.573**	+0.317
interest.n	185	7	77	0.419	**0.763**	+0.344
judgment.n	62	7	117	0.281	**0.469**	+0.188
difficulty.n	46	4	131	0.174	**0.478**	+0.304
add.v	263	6	62	0.462	**0.773**	+0.311
ask.v	261	6	46	0.282	**0.458**	+0.176
rule.v	59	3	41	0.400	**0.733**	+0.333
use.v	26	5	35	0.714	**1.000**	+0.286
simple.a	36	5	56	0.278	**0.500**	+0.222
important.a	36	5	54	0.316	**0.474**	+0.158

Table 2: Example Web1T Results from Lexical Sample

other based on local features yields a combined system that outperforms either system alone, allowing more information to be gleaned from the same small quantity of context.

Possible extensions to our system are numerous. The vocabulary could be expanded, and possibly be obtained using distributional similarity to find potential surrogates. This approach would have the advantage of yielding words that tend to occur in the same context as the target word – which is precisely what they are to be used for. Many additional approaches for refining the details of the substitute phrases are possible, such as the development of queries including multi-token wild cards, thus facilitating the use of multi-word-expressions as substitutes, rich sources of synonymy which are currently not handled by our system at all. This is particularly worthy of investigation given the weak relationship observed between the number of single-word substitutes and the performance improvement for each lemma. Opportunities also exist to explore many alternative measures other than PMI for constructing features from the corpus frequencies.

The most successful supervised WSD systems have tended to use combinations of many diverse information sources. We hope that the methods described here and the software we are releasing will help to facilitate well-understood web-scale resources to become even more useful tools in the inventory of both WSD and wider Natural Language Processing systems.

8 Acknowledgements

Many thanks to Jon Patrick, James Curran, and Matthew Honnibal for their invaluable assistance, insights and advice.

References

Eneko Agirre and David Martinez. 2004. Unsupervised WSD based on automatically retrieved examples: The importance of bias. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*, pages 189–196. Barcelona, Spain.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram corpus version 1.1. Technical report, Google Research.

Yee Seng Chan and Hwee Tou Ng. 2005. Scaling up word sense disambiguation via parallel texts. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*. Pittsburgh, Pennsylvania, USA.

Timothy Chklovski and Rada Mihalcea. 2002. Building a sense tagged corpus with open mind word expert. In *Proceedings of the Workshop on "Word Sense Disambiguation: Recent Successes and Future Directions"*, pages 116–122.

Yaacov Choueka and Serge Lusignan. 1985. Disambiguation by short contexts. *Computers and the Humanities*, 19:147–157.

Kenneth Ward Church, Willam Gale, Patrick Hanks, Donald Hindle, and Rosamund Moon. 1994. Lexical substitutability. In B. T. S. Atkins and A. Zampolli, editors, *Computational Approaches to the Lexicon*, pages 153–177. Oxford University Press.

Ido Dagan, Lillian Lee, and Fernando Pereira. 1997. Similarity-based methods for word sense disambiguation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the ACL*, pages 56–63. Madrid, Spain.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`.

Usama M. Fayyad and Keki. B. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1029. Chambery, France.

Christiane Fellbaum, editor. 1998. *Wordnet: An Electronic Lexical Database*. MIT Press.

Abraham Kaplan. 1955. An experimental study of ambiguity and context. *Mechanical Translation*, 2(2):39–46.

Claudia Leacock, Martin Chodorow, and George A. Miller. 1998. Using corpus statistics and WordNet relations for sense identification. *Computational Linguistics*, 24:147–165.

David Martinez, Eneko Agirre, and Xinglong Wang. 2006. Word relatives in context for word sense disambiguation. In *Proceedings of the 2006 2006 Australasian Language Technology Workshop (ALTW 2006)*, pages 42–50.

Rada Mihalcea. 2002. Bootstrapping large sense tagged corpora. In *Proceedings of the 3rd International Conference on Language Resources and Evaluations (LREC 2002)*. Las Palmas, Spain.

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *Proceedings of the Conference on Natural Language Learning (CoNLL 2004)*. Boston, Massachusetts.

Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The senseval-3 english lexical sample task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 25–28. Association for Computational Linguistics, Barcelona, Spain.

Rada Mihalcea and Dan Moldovan. 1999. An automatic method for generating sense tagged corpora. In *Proceedings of the American Association for Artificial Intelligence (AAAI 1999), Orlando, FL.*

George. A. Miller, Claudia. Leacock, Tengi Randee, and Ross Bunker. 1993. A semantic concordance. In *Proceedings of the 3 DARPA Workshop on Human Language Technology*, pages 303–308.

Siddharth Patwardhan, Stanjeev Banerjee, and Ted Pedersen. 2003. Using measures of semantic relatedness for word sense disambiguation. In *Computational Linguistics and Intelligent Text Processing: 4th International Conference (CICLing 2003)*, pages 241–257. Mexico City, Mexico.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In *Proceedings of the* SENSEVAL-*3 Workshop, Barcelona*, pages 41–43.

Xinglong Wang and David Martinez. 2006. Word sense disambiguation using automatically translated sense examples. In *Proceedings of the EACL Workshop on Cross Language Knowledge Induction*. Trento, Italy.

Word Sense Disambiguation in Contextual Dynamic Network Using Associative Concept Dictionary

Jun Okamoto
Keio Research Institute at SFC,
Keio University / 5322 Endo, Fujisawa-shi
2522-8520, Japan
juno@sfc.keio.ac.jp

Shun Ishizaki
Graduate School of Media and Governance,
Keio University / 5322 Endo, Fujisawa-shi
2522-8520, Japan
ishizaki@sfc.keio.ac.jp

Abstract

Many of the Japanese ideographs (Chinese characters) have a few meanings. They should be disambiguated by using their contextual information. Some of the ideographs have a few different pronunciations depending on their meanings. For example, we have an ideograph which has two pronunciations, /hitai/ and /gaku/, the former means a forehead of the human body and the latter means an amount of money or a picture frame. Conventional methods for such disambiguation have been using statistical methods with co-occurrence of words in their context. In this research, Contextual Dynamic Network Model is developed using the Associative Concept Dictionary, which includes semantic relations among concepts/words and the relations are represented with qualitative distances. In this model, an interactive activation model is used in the contextual semantic network where the activation in the network is calculated using the distances. The proposed method can disambiguate word senses by constructing dynamically the contextual semantic network according to the input text including an ambiguous word. The interactive activation model based on the contextual semantic network is evaluated by using a Japanese newspaper, Mainichi shinbun. Its effectiveness is shown by comparison with a Naive Bayes method.

1 Introduction

Word sense disambiguation is one of the difficult problems in natural language understanding by computers because it needs contextual meanings. A lot of previous works for such disambiguation have been using co-occurrence of words in their context. Several machine learning algorithms have been used based on co-occurrence information among words, such as Naive Bayes methods or Support Vector Machine (SVM) (Murata, et.al., 2003). Effectiveness of neural network approaches to the word sense disambiguation has been suggested (Woltz and Pollack 1985; Takahashi, 1995, Tsuzuki and Saito 1991). Not only the neural network architecture but also large-scale machine readable dictionaries were exploited (Veronis and Ide, 1990).

Many of the Japanese ideographs (Chinese characters) have a few meanings. They should be disambiguated by using their contextual information. In our previous paper, we proposed a method to disambiguate homographic ideographs with different pronunciations by using a Contextual Network Model (Okamoto and Ishizaki 2005). In that model, the contextual semantic network architecture is constructed based on the Associative Concept Dictionary (Okamoto and Ishizaki, 2001) including semantic relation and distance information among the concepts. In this paper, we proposed a Contextual Dynamic Network Model, where the contextual semantic network has a structure which changes dynamically depending on the input sentences or closes. If we use the concept dictionary which includes structuralized information about word senses of ambiguous words, we can disambiguate not only homographic ideogram but also various word sense disambiguations by using the Contextual Dynamic Network Model.

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 94–103,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

2 Associative Concept Dictionary

Background knowledge is crucial for computers to understand the contents of the text as well as its syntactic or shallow semantic information from input texts. The Associative Concept Dictionary (hereinafter referred to as ACD) has been built based on the results of large-scale online association experiments, which many subjects can use simultaneously in a campus network at a campus of Keio University. In these experiments, the stimulus words were fundamental ones chosen from Japanese elementary school textbooks and were presented to human subjects. The subjects were requested to give any number of associated words from the stimulus words at the seven semantic relations, hypernym, hyponym, part/material, attribute, synonym, action and situation.

Table 1 is an example of associated words by a subject where the stimulus word is "辞書" Its pronunciation is /jisyo/ and means a dictionary. The numbers in the table show the duration spent for the concept association by the subjects of which unit is second. They are durations between starting times of the association and their ending times. This temporal information is included in the results of the association experiments. For example, "publication" is a hypernym of "dictionary" while the duration is 7 second. And next associated word is "book" as hypernym of a dictionary.

Table 1: An example of association experiment (Stimulus word is "辞書" In practice, associated words are Japanese)

Semantic relations	{Associated word Response time}
Hypernym	{Publication 7} {Book 12}
Hyponym	{English dictionary 6} {Japanese dictionary 12}
Part / Material	{Entry word 18} {Word definition 33} {Page 38} {Cover page 44}
Attribute	{Difficult 6} {Easy to understand 11} {Pleasant 16}
Synonym	{Encyclopedia 17}
Action	{Read 5} {Investigate 11} {Consult 15} {Search 19} {Buy 29}
Situation	{Library 6} {Book store 27}

All of the associated concepts are, in the ACD, connected to the stimulus words with distances calculated by the following methods. The distances are obtained using a linear programming method (Okamoto and Ishizaki, 2001). It combines the following three parameters linearly: the frequency of the associated concepts (*F*), the association order of the word (*S*), and the duration spent for their association (*T*). Next, two boundary conditions are given such that one is for the shortest distance and the other for a comparatively long distance. By using Simplex method, we obtain the optimum solution for the parameter's coefficients. The first two parameters, *F* and *S*, found significant for the distance calculation and the third parameter *T* zero. Thus, the distance *D(x,y)* between concepts, *x* and *y*, is shown by the following formula:

$$D(x,y)=0.81F+0.27S, \qquad (1)$$

where $F=\frac{N}{n+\delta}$, $\delta=\frac{N}{10}-1$ (N≧10),and $S=\frac{1}{n}\sum s_i$. *N* denotes a number of the subjects who joined the experiments, and *n* denotes a number of subjects who input the associated word *y* with the same semantic relation for a given stimulus word *x*. Furthermore, δ denotes a factor introduced to limit the maximum value of *F* to 10, and s_i denotes an order of the association by a subject.

The ACD is built using the quantified distances and is organized in a hierarchical structure in terms of the hypernym and hyponym. Attribute information is used to explain the features of the given word. The synonyms of the stimulus words are also included as well as action concepts and situation concepts.

The conventional concept dictionaries (EDR, 1990) have tree structure for expressing a hierarchical one. Distances between two concepts in the dictionaries are calculated using the number of links between them, whereas the ACD has quantitative distance information between two concepts.

In the association experiment, each stimulus word has 50 subjects who were students at SFC of Keio University. The number of stimulus words is currently 1100. Total number of associated words is about 280,000. And the number of associated words, when the overlapping words are not counted, is about 64,000 words. In Figure1, "chair" is a stimulus word. And "furniture" is a higher-level concept of "chair". The numbers <1> express frequencies of subjects who gave a same associated word, <2> an average of order of association, <3>an average of response time and <4> a conceptual distances.

```
(chair                  <1>  <2>  <3>  <4>
  (hypernym              ↓    ↓    ↓    ↓
    (furniture          0.92 1.02 0.16 1.09)
    (object             0.04 2.50 0.24 7.43))
  (hyponym
    (sofa               0.48 1.92 0.42 1.96)
    (rocking-chair      0.28 1.43 0.59 2.64))
  (part/material
    (wood               0.60 1.20 0.14 1.52))
  (attribute
    (hard               0.46 1.17 0.32 1.82))
  (synonym
    (seat               0.02 1.00 0.15 8.37))
  (action
    (sit down           0.70 1.03 0.15 8.37))
  (situation
    (school             0.30 2.40 0.22 2.78)))
```

Figure1 Concept dictionary description for a stimulus word "chair" (a part of associated concepts are presented. The stimulus word and associated words are originally in Japanese)

3 Word Sense Disambiguation by Contextual Dynamic Network Model

A lot of previous works for the word sense disambiguation have been using co-occurrence of words in their context. In this research, however, we propose a Contextual Dynamic Network Model (hereinafter referred to as the CDNM) to disambiguate word senses by using an interactive activation method in contextual semantic network. The network is constructed by using the ACD, because it includes semantic relations and distance information among the words in the context. In addition, this network is not a static one but dynamic where the network structure changes depending on the context of the paragraphs, sentences, phrases or clauses which include a homographic ideograph.

By using the dynamic network, this method can disambiguate word senses based on words located near the ambiguous words.

3.1 Construction of Contextual Semantic Network

We can use not only influence of word co-occurrence in their context but also that of comparatively rich network with quantitative distances and context information for the word sense disambiguation.

The following steps show a procedure in detail for this network construction.

- Part of speech information for words (nouns, adjectives, adverbs, verbs and so on) in an input sentence is obtained from morphological analysis by using ChaSen, a Japanese morphological analysis software (Matsumoto et. al., 1999).
- A contextual semantic network is constructed by extracting semantic relations among words in the input sentence from the ACD by using the information obtained from the morphological analysis.
- When a noun in the input sentence is included as a stimulus word in the ACD, a contextual semantic network is added which starts from the noun by tracing semantic relation paths until the distance accumulated becomes a certain numerical level.
- When a word in the input sentence is found in associated words of the ACD and a stimulus word for the word is included in the nodes in the constructed contextual semantic network, the semantic relation path from the associated word to the stimulus word is assigned to the network.
- When a stimulus word is a homographic ideograph and its associated words are associated from another homographic ideograph, inhibitory links between the stimulus words and the associated words are added in the network
- Inhibitory links among stimulus words are added in the network when the stimulus words are included in a homographic ideograph.

Figure 2 shows an example of a contextual semantic network based on the ACD and a homographic ideograph in the input sentence. Let an input sentence be "The picture frame of Picasso's picture dropped from the wall, it struck my head and the forehead bled."

In this sentence, "picture frame" and "forehead" are English expressions which correspond to the homographic ideographs "額" in Japanese. This ideograph has two pronunciations /gaku/ and /hitai/. "Wood" is a part relation concept of "picture frame". "Museum" is a situation concept of "picture frame". "Eye" and "nose" are part relation concepts of "face".

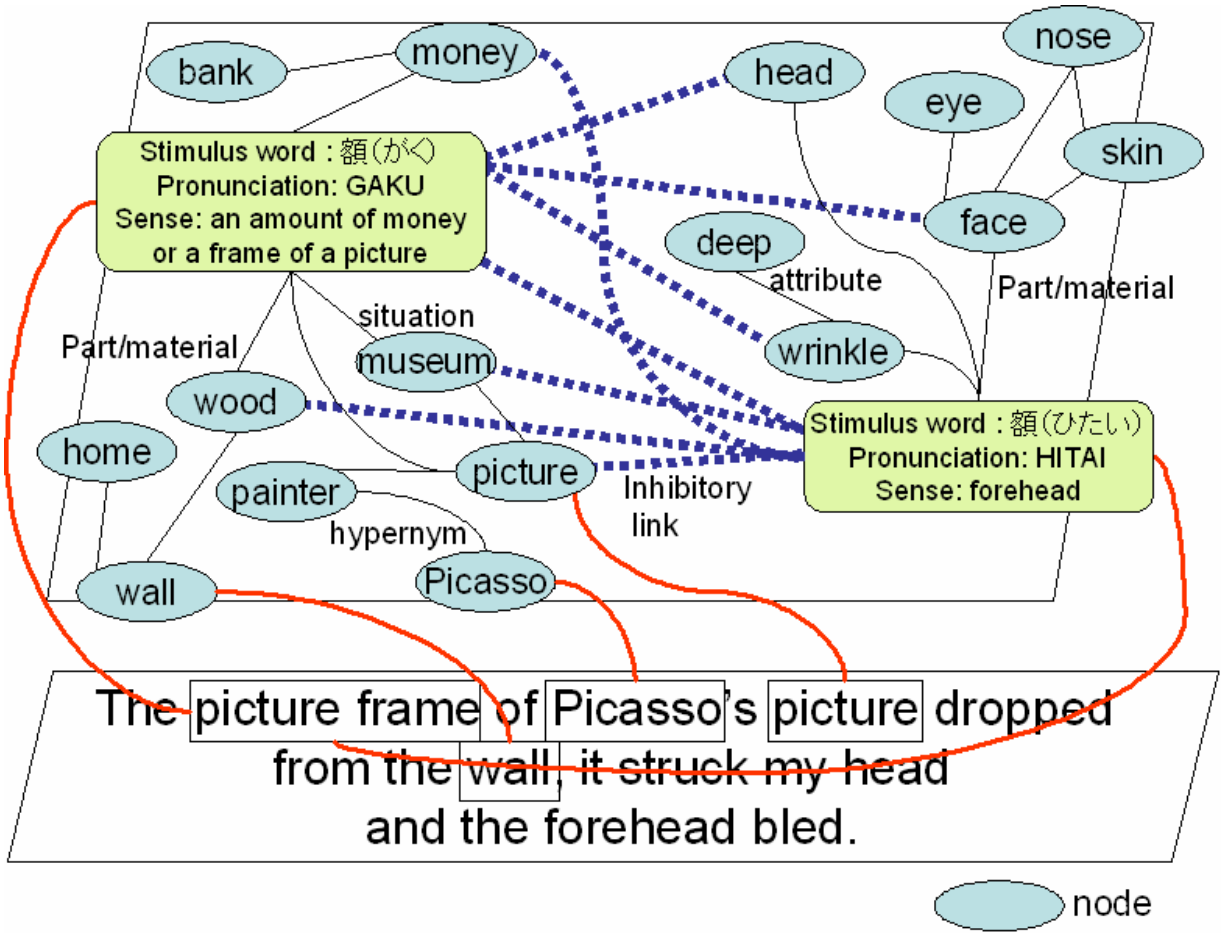


Figure 2 Example of Contextual Semantic Network based on the ACD and the words before and behind one homographic ideograph "picture frame" in an input sentence

"Deep" is an attribute concept for "wrinkle" and "wrinkle" is a part concept for "forehead." "Painter" and "face" doesn't exist in the sentence but is obtained from ACD. Oval shape nodes are added by using ACD and are connected with excitatory links. Square nodes are those of homological ideographs and are connected with inhibitory links. The thick lines connect oval nodes with the word in input sentence. The dotted lines mean inhibitory links. The associated word nodes of a homographic ideograph are connected with other homographic ideographs with inhibitory links.

3.2 Dynamic Reconstruction of Contextual Semantic Network

In this research, an input sentence is divided into multiple segments. For each segment, a contextual semantic network based on the ACD is reconstructed. Human-beings can disambiguate word senses based on two words on the left and two words on the right of the ambiguous word (Choueka and Lusignan, 1985). In our proposed method, when some homographic ideographs are in the sentence, this method can assign appropriate word senses to the given ambiguous word in the sentence depending on its contextual meaning by dividing the sentence into segments. As for the unit of the segments, we can use phrases or clauses which include one ambiguous word in some paragraphs and sentences.

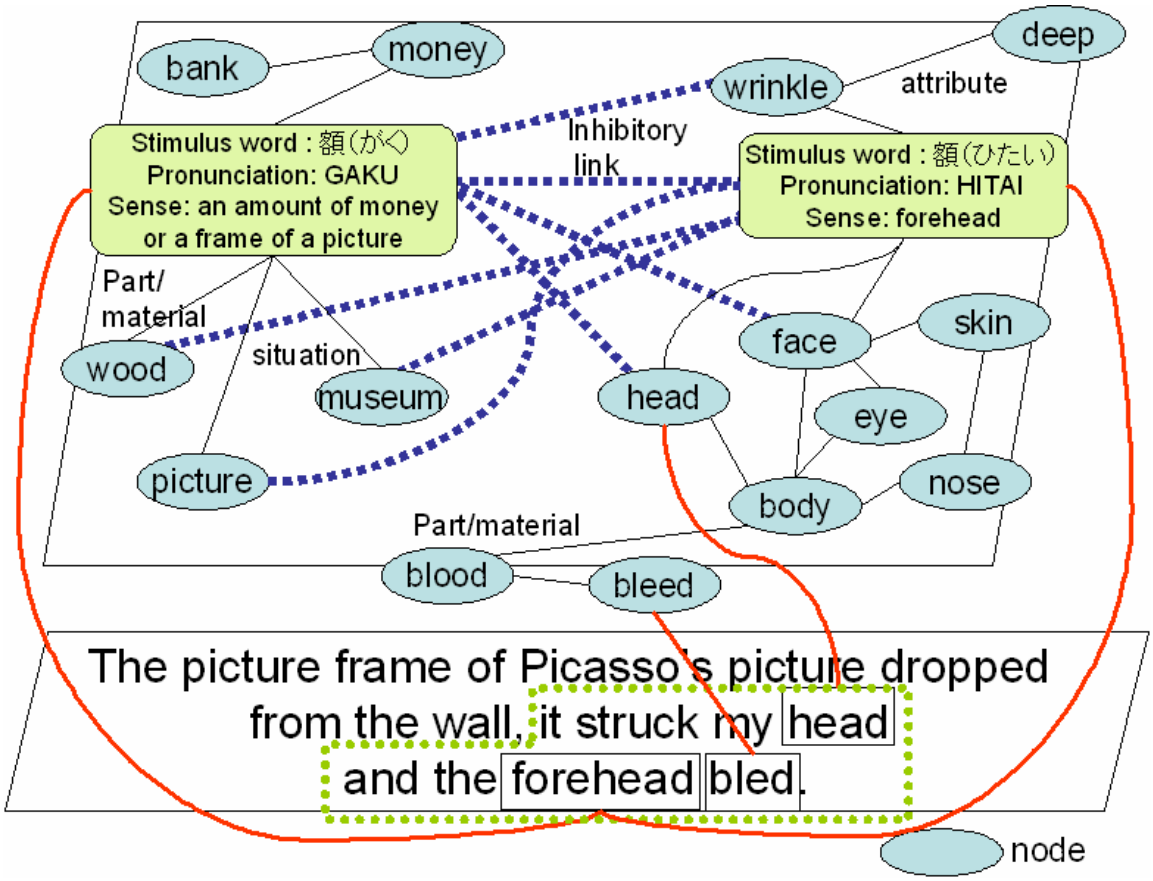


Figure 3 Example of Contextual Semantic Network based on the ACD and the words before and behind the other homographic ideograph "forehead" in an input sentence

Figure 3 shows an example of a contextual semantic network based on the ACD with the other homographic ideograph in the segments of the input sentence. New contextual semantic network is reconstructed by using the second segment in the sentence. The network includes some nodes added by using the previous segment. In addition, the activation values of the node succeed to the values in the previous segment. The polygon with broken lines in the bottom of the figure is an example of the segment in the input sentence.

3.3 Calculation of Node's Activation Value

The activation value of each node is calculated by an interactive activation model in the contextual semantic network. We define the maximum activation level as 1.0. An initial value ($a_i(0)$) is calculated by the following equation.

$$a_i(0) = 1.0 \times S_{ki}, \quad (2)$$

where 1.0 is a normalizing value for S_{ki}, which is a number of node *i* that appears in sentence k. Next, the new activation value ($a_i(t+1)$) of each node *Ni* at time *t+1* is calculated by the following equation (3).

$$a_i(t+1) = a_i(t) - \theta \cdot a_i(t) + \varepsilon_i(t) \ , \quad (3)$$

where the decay parameter θ is assumed to be 0.1 and $\varepsilon_i(t)$ expresses influence of its neighbors at time *t*. When the neighbors of a node are active, they affect the activation value of the node by excitatory or inhibitory connections, depending on a link between two nodes. Those excitatory and inhibitory influences are combined by a simple equation (4) to yield a net input to the node. Thus, $n_i(t)$ represent the net input to the node by the following the equation.

$$n_i(t) = \sum a_j(t) \big/ \alpha D_{ij} \ , \quad (4)$$

where $a_j(t)$ denotes an activation value of the node N_j connected with node N_i. α is a con-

stant weight, given by total number of links of the Contextual Semantic Network. D_{ij} denotes a distance between two nodes N_i and N_j. In this paper, The Contextual Semantic Network is constructed by tracing semantic relation paths with accumulating the distance before exceeding the value of 5.0. Therefore, the value of D_{ij} with an inhibitory link is assumed to be -5.0.

When the net input is excitatory, $n_i(t) > 0$, the effect on the node, $\varepsilon_i(t)$, is given by the following equation.

$$\varepsilon_i(t) = n_i(t)[M - a_i(t)] \quad , \quad (5)$$

where M is the maximum activation level of the node and set to 1.0. When the net input is inhibitory, $n_i(t) \le 0$, the effect of the input on the node is given by the following equation.

$$\varepsilon_i(t) = n_i(t)[a_i(t) - m] \quad , \quad (6)$$

where m is the minimum activation level of the node and set to be 0.

4 Experiment for Word Sense Disambiguation by the Proposed Method

For word sense disambiguation, we use the CDNM. This method uses an interactive activation method in a contextual semantic network. Several of Japanese ideographs in the stimulus words in the ACD have a few pronunciations. In the association experiments, such ideographs were presented as stimulus words followed with their pronunciations to avoid ambiguities.

Let an input sentence be "The picture frame of Picasso's picture dropped from the wall, it struck my head, and the forehead bled." In this sentence, "picture frame" and "forehead" are homographic ideographs in Japanese. The sentence is divided into two segments (phrases) "The picture frame of Picasso's picture dropped from the wall" and "it struck my head, and the forehead bled".

At first, we construct contextual semantic network based on the ACD for the first segment "The picture frame of Picasso's picture dropped from the wall". Figure 4 shows activation values of the homographic ideograph for by 20 time cycles. The activation values are probabilities assigned to the word at time t. This homographic ideograph is "額", which have two pronunciations /hitai/ and /gaku/, the former means a forehead and the latter means picture frame. In figure 4, the horizontal axis represents time, and the vertical axis represents activation values.

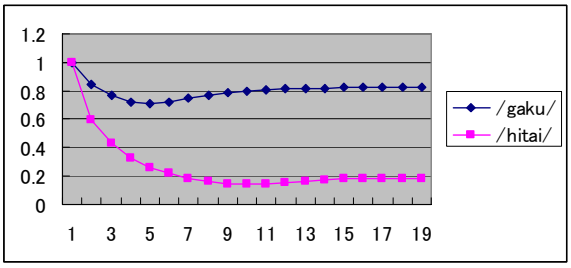


Figure 4 The time course of activation values of selected two nodes "額（がく）/gaku/, picture frame" and "額（ひたい） /hoitai/, forehead" for the first segment

The activation values of "額（がく）/gaku/, picture frame" are higher than the activation values of "額（ひたい）/hitai/, forehead".

Next, we construct a contextual semantic network based on the ACD for the second segment "it struck my head, and the forehead bled". Figure 5 shows an output activation values about homographic ideograph for the simulation through the 20 time cycles. In figure 5, the horizontal axis represents time, and the vertical axis represents activation values.

The activation values of "額（ひたい）/hitai/, forehead" are higher than the activation values of "額（がく）/gaku/, picture frame". We can disambiguate word senses by using words near the homographic ideograph and the interactive activation model.

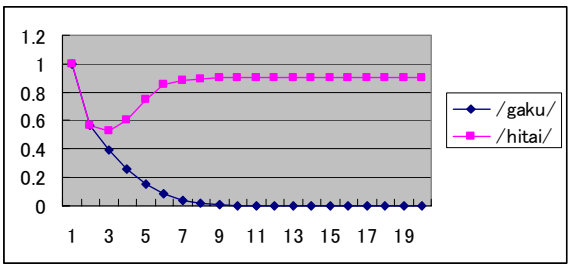


Figure 5 The time course of activation values of selected two nodes "額（がく）/gaku/, picture frame" and "額（ひたい）/hitai/, forehead" for second segment

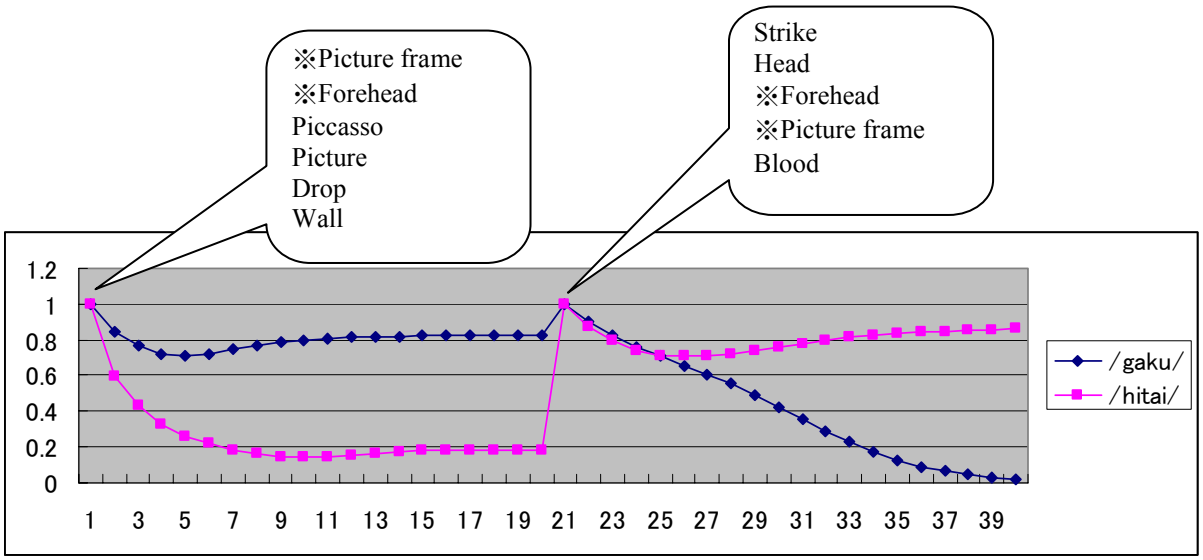


Figure 6 The time course of activation values of selected two nodes "額（がく）/gaku/, picture frame" and "額（ひたい）/hitai/, forehead" for input the sentence "The picture frame of Picasso's picture dropped from the wall, it struck my head, and the forehead bled."

Next, we disambiguate word senses by using the CDNM inputting the sentence "The picture frame of Picasso's picture dropped from the wall, it struck my head, and the forehead bled." The two segments are connected and the activation values of the second segment succeed to the values in the first segment. The initial values of the nodes in the second segment are set to 1.0.

Figure 6 shows activation values of homographic ideograph in the simulation through the 20 time cycles each segment. In figure 6, the horizontal axis represents time, and the vertical axis represents activation values. Several words in two words balloons are speech information for words (nouns, adjectives, adverbs, verbs and so on) in each input segment. In the words balloons, "Picture frame" and "Forehead" are homographic ideograms "額（がく）/gaku/" and "額（ひたい）/hitai/" sequentially.

The contextual semantic network is constructed by using the first segment (picture frame, forehead, Piccasso, picture, drop and wall) in the sentence when the time equals to 1 on horizontal axis. At this point, the initial values of their nodes are set to 1.0. Next new contextual semantic network is reconstructed by using the second segment (strike, head, forehead, picture frame and blood) in the sentence when the time equals to 21 on horizontal axis. The network includes some nodes added by using the previous segment. Here the activation values of the nodes succeed to the values in the first segment and the initial values of their nodes in the second segment are set to 1.0.

The activation values of "額（がく）/gaku/, picture frame" are higher than the activation value of "額（ひたい）/hitai/, forehead" for first segment. Next, the activation values of "額（ひたい）/hitai/" are higher than the activation values of "額（がく）/gaku/, picture frame" for the second segment.

Table 2 shows top ten activation node value for the first segment "The picture frame of Picasso's picture dropped from the wall". "Wood" is a material concept of "picture frame". "Room" is a situation concept of "picture". The activation node values concerning "wood" and "picture" are comparatively high.

Table 2 Top ten activation node values for first segment

Nodes	Activation values
木 -Wood -	0.94
紙 - Paper -	0.92
顔 - Face -	0.91
植物 - Plant -	0.91
壁 - Wall -	0.89
絵本 - Picture book -	0.87
葉 - Leaf -	0.87
絵 - Picture -	0.87
部屋 - Room -	0.87
カレンダー - Calendar -	0.86
皮膚 - Skin -	0.84
森 - Forest -	0.84
根 - Root -	0.84
額（がく） - Picture frame -	0.83
図 - Figure -	0.83

Table3 shows top ten activation node value for second segment "it struck my head, and the forehead bled". "Face" is situation concept of "forehead". "Skin" is part concept of "forehead". "Wounded" is situation concept of "blood". The activation values concerning "face" and "head" are comparatively high. It is thought that the activation value of "wood" is high because the activation value of the "living-thing" is comparatively high.

Table 3 Top ten activation node values for second segment

Nodes	Activation values
木 -Wood -	0.94
血 - Blood -	0.93
顔 - Face -	0.93
植物 - Plant -	0.91
頭 - Head -	0.91
皮膚 - Skin -	0.89
人間 - Human-beings -	0.89
葉 - Leaf -	0.87
額（ひたい） - Forehead -	0.86
森 - Forest -	0.85
傷 - Wounded -	0.84
生物 - Living-thing -	0.84
根 - Root -	0.84
目 - Eye -	0.83
口 - Mouth -	0.81

5 Comparison of Proposed Method and Conventional Method

5.1 Naive Bayes Method for Word Sense Disambiguation

We use a framework of multinomial Naive Bayes text classification for word sense disambiguation. Let s_i, i = 1, 2,…, m, denote a word sense of a homographic ideogram. S is a set of the word senses. Let w_j, j = 1, 2,…, n, denote words appear in the paragraph. We obtain the optimum word sense s which maximizes $P(s_i \mid w_1,...,w_n)$ by the following functions.

$$\begin{aligned} s &= \arg\max_{s_i \in S} P(s_i \mid w_1,\cdots,w_n) \\ &= \arg\max_{s_i \in S} P(w_1,\cdots,w_n \mid s_i)P(s_i) \end{aligned},$$

where $P(s_i)$ is a number of paragraphs including s_i divided by total number of the paragraphs. We have a Naive Bayes assumption that words surrounding w_j is independent each other:

$$P(w_1,\cdots,w_n \mid s_i) = \prod_{j=1}^{n} P(w_j \mid s_i) .$$

We can determine a probability that it belongs to class s_i by the Bayes' rule:

$$s = \arg\max_{s_i \in S} P(s_i) \prod_{j=1}^{n} P(w_j \mid s_t) .$$

Next, we apply the Jeffreys-Perks law (Good, 1965) to solve zero frequency problems. Naive Bayes is a simple and effective method in the statistical machine learning techniques. Despite of its simplicity, this method is effective and often applied to word sense disambiguation.

5.2 Experiment Evaluation and Results

A pair of training and test data from large corpora is used to evaluate the interactive activation model based on the contextual semantic network. The accuracy rate of homographic ideogram's correct pronunciation in all the test data are compared among the interactive activation model and the Naive Bayes method. Several of Japanese ideographs in the stimulus words in ACD have a few pronunciations. In the association experiments, such ideographs were presented as stimulus words with pronunciations to avoid ambiguities by subjects.

A lot of data sets are extracted for the pair of training data and test data including homo-

graphic ideograms “額” and “金” from a Japanese newspaper, Mainichi Shinbun (from 1993 to 1995). These data sets include ten words on the left and ten words on the right of the ambiguous words as shown in Table 4. The number of data sets including an ideograph “額” is 1537, and the number of data sets including an ideograph “金” is 3672. The ideograph “金” has two pronunciations /kane/ and /kin/ just like the ideograms “額”, the former means money and the latter means gold, gold medal, Friday or a piece of the Japanese chess.

Next, we labeled correct pronunciations for the homographic ideograms in all the training and test data. Pairs of training data (about 95% of data sets) and test data (about 5% of data sets) are used to evaluate the performance of the proposed method. The evaluation for a word sense disambiguation is designed to check the effectiveness of the CDNM. We use data sets for homographic ideogram “額” which means “forehead” or “picture frame”, and use ones for homographic ideogram “金” which means “gold” or “money”

Table 4 Comparison of test results among the two methods. NB: Naive Bayes Method

	neighborhood words	Our method	NB
An ideograph “額” pronunciation as /hitai/ and /gaku/	10 words	90.4%	80.1 %
An ideograph “金” pronunciation as /kin/ and /kane/	10 words	86%	90.6 %

Table 4 shows accuracy rates of disambiguation for “額” or “金” in test data. The neighborhood words are words on the left or right of an ambiguous word. Our method shows the best score for the two homographs as the correct pronunciations’ ratio in all test data of homographic ideogram “額”. About an homographic ideogram “金”, the data sets of pronunciation /kin/ include a lot of idioms. For example “金の卵 /kin no tamago/” that means a “Golden boy”. Therefore NB shows best score for homographic as the correct pronunciations’ ratio in all test data of homographic ideogram “金” because there are a lot of idioms in test data sets which include neighborhood words of ambiguous words.

It is difficult to disambiguate word sense by using the proposed method. However the correct pronunciations’ ratio is 98% about the test data of pronunciation /kane/.

6 Future Work

In this research, we proposed a method for disambiguation of pronunciations of homographic ideographs by using the CDNM. We made computer simulations to show effectiveness of this model and consistency with empirical data.

The ideograph “額” has two major senses when its pronunciation is /gaku/. One is an amount of money and the other is a picture frame. It is necessary to carry out more experiments to disambiguate the two meanings as well as the two pronunciations. In addition, if we use the concept dictionary which includes structurized information about word senses of ambiguous words, we can disambiguate not only homographic ideogram but also various word senses by using the Contextual Dynamic Network Model.

Acknowledgement

We wish to express our gratitude to the students at SFC, Keio University who were very helpful for the association experiments, and also to the member of Ishizaki Laboratory who helped us to construct and modify the Associative Concept Dictionary.

References

Choueka, Y. and Lusifnan, S., “Disambiguation by short contexts”, Computers and the humanities, Vol. 19, pp 147-157, 1985.

EDR, “EDR Electronic Dictionary Technical Guide”, Japan Electronic Dictionary Research Institute, Ltd., 1990.

Matsumoto, Y., Kitauchi, A., Yamashita, T., Hirano, Y., Matsuda, H. and Asahara, M., “Japanese Morphological Analysis System ChaSen Manual version 2.0 Manual 2nd edition” (in Japanese)., NAIST Technical Report, NAIST-IS-TR99009 Nara, Institute of Science and Technology, 1999.

McCelland J.L. and Rumelhart D.E., “An Interactive Activation Model of Context Effects in Letter Perception: Part 1. An Account of Basic Findings”, Psychological Rev., Vol.88, No.5, pp375-407,1981.

Murata, M., Utiyama, M., Uchimoto, K., Ma, Q. and Isahara, H., “CRL at Japanese dictionary-based task of SENSAVAL-2 -- Comparison of various

types of machine learning methods and features in Japanese word sense disambiguation --" (in Japanese)., Journal of NLP Vol.10 No.3, 2003.

Okamoto, J. and Ishizaki, S., "Construction of Associative Concept Dictionary with Distance Information, and Comparison with Electronic Concept Dictionary" (in Japanese)., Journal of NLP Vol.8 No.4, 2001.

Okamoto, J. and Ishizaki, S., "Word Sense Disambiguation Using Contextual Semantic Network Using Associative Concept Dictionary", SNLP 2005, pp205-210, 2005.

Takahashi, N., "Lexical Disambiguation by Layered Neural Networks" (in Japanese)., IPSJ Journal, Vol.36 No.9, 1995.

Veronis, J. and Ide, N. M., "Word Sense Disambiguation with Very Large Neural Networks Extracted from Machine Readable Dictionaries"., Coling '90, 1990.

Waltz, D.L. and Pollack, J.B., "Massively parallel parsing: A strongly interactive model of natural language interpretation", Cognitive Science, Vol.9,pp51-74,1985.

Large-Scale Supervised Models for Noun Phrase Bracketing

David Vadas and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{dvadas1, james}@it.usyd.edu.au

Abstract

Interpreting the structure of noun phrases (NPs) is important for many Natural Language Processing (NLP) tasks. This work extends the state-of-the-art in NP bracketing by: creating supervised models trained on a large annotated corpus; applying these to longer, more complex NPs; and using the resulting system to improve the output of the Bikel (2004) parser.

Using a large corpus of manually annotated Penn Treebank NPs we have developed a supervised model that brackets simple NPs with 93.01% F-score. We extend the evaluation to include longer, more complex NPs that are rarely dealt with in the literature, attaining 91.44% F-score. Finally, we implement a post-processing module that brackets NPs identified by the Bikel (2004) parser, which outperforms the parser itself by 8.13% F-score.

1 Introduction

Noun phrase (NP) bracketing is a requirement for the syntactic and semantic analysis of NPs. In the literature, e.g. Marcus (1980, p253) and Lauer (1995), the task is generally framed as follows: given a 3 word noun phrase like those below, decide whether it is left branching (1) or right branching (2).

```
((crude oil) prices)        (1)
(world (oil prices))        (2)
```

NP bracketing is crucial for many Natural Language Processing (NLP) tasks. For example, question answering (QA) and anaphora resolution both require (potentially nested) candidate NPs, typically identified using a parser. If the answer or antecedent is not the complete NP, e.g. `crude oil` above, then it cannot be found.

Unfortunately, internal NP bracketing is not identified by parsers trained on the Penn Treebank, e.g. Collins (1999), because the Penn Treebank (Marcus et al., 1993), has flat NP structure as shown below:

```
(NP (NNP Air) (NNP Force) (NN contract))
```

Instead, researchers have attempted to solve the NP bracketing problem with unsupervised methods based on statistics from unannotated corpora (Lauer, 1995) or web hit counts (Lapata and Keller, 2004; Nakov and Hearst, 2005a).

The main contributions of this paper are large-scale supervised models trained on bracketed NPs from the entire Penn Treebank. The corpus also provides a large and representative test set. Using this data we significantly outperform previous approaches on the NP bracketing task. By incorporating a wide range of existing and novel features into the model, we increase performance by 8.87% in F-score over our best unsupervised system.

Further, we experiment with bracketing longer, *complex* NPs, by including NPs longer than three words, and which include non-nominal parts of speech (such as adjective, determiner, etc). These have been largely neglected in the literature, in contrast to *simple* NPs, i.e. those that are three words long and contain only nouns. We reimplemented the bracketing algorithm of Barker (1998) for complex noun phrases using the sophisticated supervised model we used for simple NPs. Our system achieves a performance of 91.44% F-score on matched brackets.

Finally, we apply these supervised models to the output of the Bikel (2004) parser. Our post-processor achieves an F-score of 77.27%, compared to parser output baseline of 69.14%. This is the first work to demonstrate that both simple and complex base-NPs can be bracketed with a high level of accuracy using supervised models.

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 104–112,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

2 Background

NP bracketing is similar to chunking (Ramshaw and Marcus, 1995), as both tasks aim to identify NP structure. Recursive NP bracketing, as in the CoNLL 1999 shared task and as performed by Daumé III and Marcu (2004) is closer still. However both these task are strictly less difficult than NP bracketing as defined in this paper, as they do not attempt to recover the full extent of sub-NP structure. This is in part because gold-standard annotations for this task have not been available in the past.

A basic method for solving the simple NP bracketing task was first described in Marcus (1980). This *adjacency* model compares the semantic association of words 1–2 to that between words 2–3. If the former is more likely, then the compound is left branching, otherwise it is right branching. Various methods of measuring the semantic association between a pair of words have been proposed for NP bracketing (Pustejovsky et al., 1993; Resnik, 1993) but they all depend on counting occurrences of bigrams in some corpus. Metrics such as χ^2 and mutual information can be used instead of the raw counts, and have been shown to perform well (Nakov and Hearst, 2005a).

Lauer (1995) proposes a new variation: the *dependency* model. In this case, we compare the semantic association of words 1–2 to that of words 1–3. This change is motivated by the dependencies that arise from the structure of the NP. We would expect a dependency between words 2–3 whether the compound was left or right branching, so there is no reason to analyse it.

Lauer (1995) demonstrated the superior performance of the dependency model using a test set of 244 (216 unique) noun compounds drawn from Grolier's encyclopedia. This data has been used to evaluate most research since. Lauer uses Roget's thesaurus to smooth words into semantic classes, and achieves 80.7% accuracy.

Lapata and Keller (2004) derive bigram probability estimates from web counts, and unlike Lauer, they use no smoothing beyond the lexical level, achieving 78.7% accuracy. Nakov and Hearst (2005a) also use web counts, but incorporate additional counts from several variations on simple bigram queries, including queries for the pairs of words concatenated or joined by a hyphen. This results in an impressive 89.3% accuracy.

There have also been attempts to solve this task using supervised methods, even though the lack of gold-standard data makes this difficult. Girju et al. (2005) annotate a training set of 362 NPs drawn from WSJ text and use it to train a decision tree classifier, achieving 73.1% accuracy. Their feature set is made up of semantic information from WordNet. When they shuffled their data with Lauer's to create a new test and training split, their accuracy increased to 83.1%, which may be a result of the ∼10% duplication in Lauer's test set.

Barker (1998) describes an algorithm for bracketing a complex NP (described in Section 5) that reduces the problem to making a number of decisions on 3 word NPs. This system attains 62% and 65% accuracy on two different data sets for the simple NPs that are processed during the algorithm. Our supervised method performs as high as 96.19% on simple NPs, and so the performance for complex NPs in our implementation of Barker increases dramatically as well.

We have experimented with most of the features described in the literature: semantics; counts drawn from search engines and from a very large corpus of 1 trillion words; and additional lexical queries. We also implement novel features based on the NP's context and the local POS and NER tags, and combine them all using a maximum entropy model.

3 Data

Vadas and Curran (2007) manually annotated all noun phrases in the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993). The annotation process involved inserting NML and/or JJP nodes to define each NP's internal structure, as shown below:

```
(NP
  (NML (NNP Air) (NNP Force))
  (NN contract))
```

The agreement for the annotations was 98.52% (exact NP match) on WSJ Section 23 after discussion between the two annotators.

We now use the Vadas and Curran data to extract simple NPs from the Penn Treebank. If the last three children of an NP are nouns, then they became an example in our data set. Lauer (1995) used a similar approach of collecting all three noun sequences and then manually filtering them from Grolier's encyclopedia.

If the first and second words are bracketed, then it is left branching, otherwise it is right branching. The annotation scheme does not currently

#	H	SEQUENCE	EXAMPLE
1053	0.13	(JJ JJ NNS)	big red cars
1353	0.60	(DT JJ NN NN)	the high interest rate
1480	0.55	(JJ NN NNS)	high interest rates
1761	1.00	(NNP NNP NNP)	John A. Smith
1793	1.00	((NNP NNP) NNP)	John Smith Co.

Table 1: Complex NP POS tag sequences

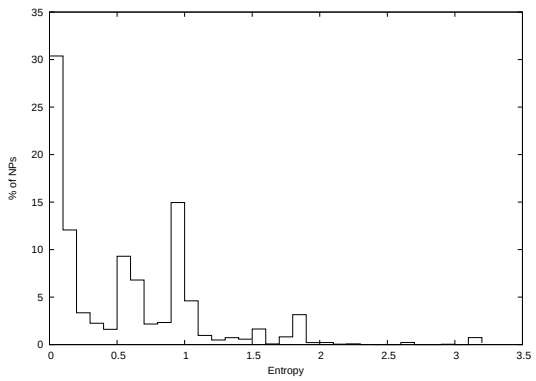


Figure 1: Entropy of NP POS tag sequences

mark flat base-NPs. Instead, all structure is either marked as left branching or implicitly right branching.

Note that because we are only looking at the right-most part of the NP, we know that we are not extracting any incomplete NPs. Finally, we remove examples where each word has the same NER tag, ignoring many of the flat base-NP cases such as *John A. Smith*.

This process results in 5582 three word NPs for bracketing, which is an order of magnitude larger than all previous data sets. Previous researchers have typically used Lauer's set (244 NPs) or created their own small set (∼500 NPs at most). This new, much larger data set means that we can carry out sophisticated machine learning effectively, rather than using unsupervised methods.

We have also extracted another even larger data set of complex NPs, for the experiments in Section 5. For this set we retrieve an example for each base-NP of length three or more in the Penn Treebank. Some common POS tag sequences (e.g. initial determiner and final possessive) are unambiguous in three word NPs, and so we remove these cases. This leaves 36584 instances in our data set, which is two orders of magnitude larger than any that has been created previously.

Table 1 shows the most common POS tag sequences in our complex NP data set. The entropy H of the distribution of bracketings for the POS tag sequences gives an indication of the difficulty of the task. Larger entropy means that the sequence is more ambiguous, either because there are many bracketing alternatives to choose from or because the alternatives are close to equally likely.

Figure 1 shows a histogram of the entropy distribution across POS tag sequences. While 24% of all sequences have a single bracketing, the majority of sequences are ambiguous. There is a spike just below 1 bit, where sequences have two almost equally likely bracketings. This demonstrates that complex NP bracketing is far from a trivial task.

COUNTS	MODEL	LAUER	PTB
Google	χ^2 adjacency	72.95	79.86
	χ^2 dependency	76.23	70.15
	Voters	83.61	83.98
MSN	χ^2 adjacency	72.13	79.60
	χ^2 dependency	74.59	69.76
	Voters	81.97	83.27
Web 1T	χ^2 adjacency	74.59	80.40
	χ^2 dependency	82.38	70.62
	Voters	83.20	81.60

Table 2: Unsupervised results for simple NPs

4 Experiments on Simple NPs

4.1 Unsupervised Approach

For these unsupervised experiments, we used all 5582 simple NPs as test data, and our hit counts come from search engine queries on the web, following Lapata and Keller (2004). We implement both the adjacency and dependency models, using the raw counts, the bigram probability $P(w_i, w_j|w_j)$, and the χ^2 measure. We also retrieved hit counts for a number of other variations that were proposed by Nakov and Hearst (2005a), such as the bigram joined by a hyphen to form a single token, or with a possessive marker.

The results using the best performing metric, χ^2, are shown in Table 2. The voting result comes from choosing a subset of the web query variations, each of which votes for left or right branching. Following Nakov and Hearst (2005a) the results shown for these models are optimal for each test set, with no development set. When optimised on a held-out set, the results varied significantly, and were generally worse than choosing the best individual model. Our best result on Lauer's data, 83.6%, is lower than Nakov and Hearst's 89.3%. However, we haven't implemented paraphrases or the morphological queries since they require a huge number of searches.

COUNT SOURCES	LAUER	PTB
Google / MSN	0.97	0.72
Google / Web 1T	0.92	0.60
MSN / Web 1T	0.93	0.81

Table 3: Correlation between counts

Our counts come from three different sources: Google and MSN search engine hit counts, and from the Google Web 1T corpus (Brants and Franz, 2006), which contains n-gram counts collected from 1 trillion words of web text. It is interesting to see how much the results vary, which indicates how sensitive this unsupervised method is to the counts it is based upon. Table 3 shows the correlation between the counts sources on the two data sets. This highlights the large degree of variability between our three sources of web counts especially on the larger Penn Treebank set. However, Nakov and Hearst (2005b) found that this variation did not have a significant impact on the performance of their NP bracketing system.

4.2 Supervised Models

Supervised models typically outperform unsupervised models for most NLP tasks. For NP bracketing, the small quantity of gold-standard data has meant that few supervised models have been implemented, and those that have been, performed poorly. With our new, significantly larger data set covering the Penn Treebank, we have built the first wide-coverage supervised NP bracketer.

We use the MegaM Maximum Entropy classifier (Daumé III, 2004) and discretise non-binary features using Fayyad and Irani's (1993) algorithm. Maximum entropy models allow diverse and overlapping features to be incorporated in a principled manner.

The initial features in our model were the counts used in the unsupervised experiments. For each query in Nakov and Hearst's extended set, we get a left count, a right count, and a binary feature representing the left or right vote. We also included the corresponding probability and χ^2 metrics for the original queries (i.e. not including the extended Nakov and Hearst queries).

All of these features were defined over both the adjacency and dependency representations. The counts themselves come from Google and the Web 1T corpus, with separate features for each source. This first supervised model has 860 features in total, all based on the unsupervised counts.

MODEL	F-SCORE
Unsupervised, voting	84.14
All unsupervised features	90.59
All supervised features	90.59
−Web searches	91.94
−Web 1T corpus	93.01
−Lexical	92.20
−POS	92.56
−NER	92.20
−Context sentence	92.74
−Context window	92.65
−Semantic	92.74
All features	**93.01**

Table 4: Supervised results

The results on our Penn Treebank test set are shown in Table 4. The unsupervised voting system is included for comparison purposes. We can see that the supervised model with unsupervised features outperforms the unsupervised voting model by 6.45%. This improvement comes from the supervised model's ability to weight the individual contributions of all of the unsupervised counts from Google and the Web 1T corpus.

One of the main advantages of using a maximum entropy classifier is that we can easily incorporate a wide range of features in the model. We have added lexical features for all unigrams, bigrams and the trigram with in the NP. All of these features are labelled with the position of the n-gram within the NP.

Since we are bracketing NPs *in situ* rather than stand-alone NPs (like Lauer) we can exploit the context around the NP as well. To do this we added bag-of-word features for all words in the surrounding sentence, and well as specific features for a two-word window around the NP.

As we are using the Penn Treebank, we have access to gold-standard POS and NER tags (the latter from the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005)). For every n-gram and context window feature, we also added generalised features by replacing the words with their POS and NER tags. POS tags are included even though all the words in the NP are nouns for these simple NP experiments, as they may be proper and/or plural.

Finally, we incorporate semantic information from WordNet (Fellbaum, 1998). For each sense of each word in the NP, we extract a semantic feature for its synset, and also the synset of each of

its hypernyms up to the WordNet root. These additional feature types increase the number of features in the maximum entropy classifier to 88,568.

Table 4 shows the results for a model using only the supervised features, and a combination of the supervised and unsupervised features. It also presents a subtractive analysis. The supervised and unsupervised features each contribute equally, while removing the Web 1T counts does not decrease performance at all, probably because they overlap significantly with the web searches.

Of the supervised features, the lexical and NER are most important but all make a positive contribution. Our best performance of 93.01% F-score comes when using all features.

5 Experiments on Complex NPs

Up until now, all of our experiments (and almost all results from the literature) have only focused on NPs that consist of exactly three nouns (noun compound bracketing). This is a simplification of the actual problem, where longer NPs with higher levels of ambiguity make finding the correct bracketing significantly harder. Adjectives, determiners and other non-nominal parts of speech also complicate the task.

Barker (1998) describes a method for bracketing these complex NPs, by reducing the problem to a series of three word bracketing decisions using a sliding window. These decisions are made using the techniques described above for simple NPs. Barker's algorithm is shown in Figure 2.

When a pair of words are bracketed, the head is chosen to represent the phrase and remains in the window. The window then expands one word to the right, unless it is already rightmost in which case it grows to the left. We use the standard head-finding rules of Collins (1999).

The complex NP data set is extracted from all of the Penn Treebank NPs annotated by Vadas and Curran (2007). We have extracted 36,584 complex NPs, which we split in a 9:1 ratio, giving 32,925 and 3659 training and test examples respectively.

5.1 Evaluation measures

The complex NP results are evaluated using several measures. Matching brackets is the standard Parseval evaluation method (Sekine and Collins, 1997). We also report implicit matching brackets which treats the structure as a binary tree, where implicit brackets are assumed to be right branching. Exact match measures the percentage of complex NPs that are entirely correct, and simple NPs measures the model's performance on the 3 word NPs that are processed during Barker's algorithm.

---

w is the current position of the window

1. w initially covers the last 3 words
2. Bracket the words in w
3. If w is left branching:
   (a) If w is in the left-most position, bracket the left two words in w
   (b) Otherwise, move w one word to the left. We cannot left bracket yet, because it might be X)Y)Z, not (XY)Z
4. If w is right branching:
   (a) Bracket the right two words in w
5. If there are only two words left, then finish. Otherwise, go to step 2

Figure 2: Barker's (1998) NP bracketing algorithm

Matched brackets is a fairly tough evaluation, as the more easily identified flat NPs are not taken into account. For example, a baseline of always choosing right branching will score 0.0, as no explicit brackets are needed.

We only report F-score for implicit brackets, as there is a set number of brackets dependent on the length of the word, and so precision and recall are always equal. Finally, note that the simple NPs are different for each model, as the next three word NP to bracket depends on the decisions made previously for this complex NP, so the numbers are not directly comparable.

5.2 Complex NP Results

Our first experiment implemented Barker's algorithm, using only the χ^2 dependency and adjacency methods. We only use counts from the Web 1T corpus, since performing web searches has become impractical with the increased data set size and NP length. The difficulty of complex NP bracketing can be seen by the drop in performance using these simple approaches, e.g. from 80.40% to to 58.41% for the adjacency model.

We next applied our supervised approach to complex NPs. This is more complicated now as we need to extract a training set of three word win-

MODEL	MATCHED BRACKETS			IMPLICIT	EXACT	SIMPLE
	P	R	F			
Baseline – right branching	0.00	0.00	0.00	62.25	54.66	62.60
χ^2 Dependency	23.47	52.26	32.40	40.91	32.66	52.53
χ^2 Adjacency	24.47	43.72	31.37	49.34	35.86	58.41
All features	88.10	88.19	88.15	93.59	90.52	94.83
−Web 1T corpus	88.50	86.51	87.49	93.13	89.97	94.43
−Lexical	87.77	87.68	87.73	93.11	90.11	94.40
−POS	86.41	86.72	86.56	92.72	89.29	94.16
−NER	87.76	87.23	87.49	93.23	90.13	94.53
−Context sentence	89.58	88.85	89.22	93.98	91.25	95.24
−Context window	88.27	88.45	88.36	93.62	90.71	94.82
−Semantic	87.46	87.33	87.39	92.96	89.75	94.49
−Non-head words	85.34	85.90	85.62	92.58	89.04	94.11
−Border words	88.15	87.79	87.97	93.48	90.74	94.79
−POS tag sequence rule	88.33	87.79	88.06	93.48	90.41	94.73
Best (500 iterations)	92.41	90.48	91.44	95.16	93.03	96.19

Table 5: Complex NP results

dows from the complex NPs. To do this, we run Barker's algorithm on the 32,925 complex NPs. At each decision point, we bracket left or right according to the gold standard, and store the three word window as a training example. This process produces 77,900 training examples.

We experiment with the features used for simple NPs as well as some novel features. Firstly, we added features encoding the non-head words when the window already contains a bracket, which increased performance by 2.5% on matched bracket F-score. Secondly, we add the bigram of the words on the NP border, that is, where it overlaps with the context. Lastly, we extracted POS tag sequences with low entropy, i.e. are quite unambiguous, and have a feature explicitly encoding their most common branching in the training data.

The results are shown in Table 5. The supervised methods significantly outperform the unsupervised methods, with a matched brackets F-score comparable to the Collins (1999) parser's overall performance. We carried out a subtractive analysis of the feature types, finding that context sentence has a negative impact on performance, and our best result comes from removing it. We ran the final experiment using 500 rather than the default 100 iterations in MegaM, to allow the estimation to converge.

5.3 Parser Post-Processing

This final set of experiments use the complex NP models as a post-processing step for a parser.

EVALUATING	P	R	F
All brackets	88.63	88.29	88.46
NML JJP	77.38	62.49	69.14
Base-NP NML JJP	48.09	56.30	51.87

Table 6: Bikel (2004) parser performance

EVALUATING	P	R	F
All brackets	88.57	88.64	88.60
NML JJP	77.31	77.23	77.27
Base-NP NML JJP	88.78	78.97	83.59

Table 7: Post-processor performance

Since our data is simply bracketed structure for the Penn Treebank, we can train a parser model and evaluate it on NP structure. We train Bikel's (2004) implementation of Collins' (1999) parser on Sections 02–21 of the Penn Treebank, and test on all sentences in Section 00.

Table 6 displays the parser's performance. It originally achieved 88.92% F-score on all constituents, but training with the NP bracketing structure makes the task more difficult, dropping the F-score to 88.46%. We are particularly interested in the performance on just the NML and JJP nodes, and on the base-NPs that we will bracket. These figures form the baseline that we aim to improve on with our post-processing module.

There are two factors that reduce the upper bound on post-processing performance. Firstly, the post-processor only brackets NPs identified by

MODEL	MATCHED BRACKETS			IMPLICIT	EXACT	SIMPLE
	P	R	F			
χ^2 Dependency	15.47	51.71	23.81	40.32	36.47	46.24
χ^2 Adjacency	16.21	47.66	24.19	46.82	39.03	51.49
All features	73.85	85.17	79.11	90.96	90.62	91.94
−Web 1T corpus	76.06	81.75	78.80	91.14	90.68	92.07
−Lexical	71.46	79.97	75.48	89.79	89.18	90.91
−POS	74.66	82.51	78.39	91.14	90.48	92.13
−NER	76.55	84.41	80.29	91.85	91.51	92.82
−Context sentence	76.11	84.79	80.22	91.69	91.31	92.63
−Context window	75.60	84.41	79.76	91.58	91.15	92.50
−Semantic	75.86	83.65	79.57	91.44	91.05	92.35
−Non-head words	72.96	82.76	77.55	91.40	90.35	92.40
−Border words	76.57	83.65	79.95	91.49	91.21	92.40
−POS tag sequence rule	76.18	83.90	79.86	91.62	91.18	92.58
−Parser	75.23	83.90	79.33	91.37	90.98	92.38
Best	77.70	83.90	80.68	92.06	91.45	92.95
Best (500 iterations)	79.40	84.54	81.89	92.75	92.35	93.66

Table 8: Complex NPs for parsing results

the Bikel parser, which are incorrect in approximately 10% of cases. Secondly, in the Vadas and Curran (2007) annotation scheme, NML and JJP nodes may appear in any NP, not just base-NPs.

Our method only applies to base-NPs, so we cannot achieve 100%, as NML and JJP nodes dominated by non-base-NPs cannot be corrected. The parser achieves 51.87% on the base-NP subset of nodes. If we were 100% correct in this subset, then the parser would still only achieve 89.06% F-score over all NML and JJP nodes.

We train the complex NP bracketer on gold-standard NPs from Sections 02–21, extracting 59,247 complex NPs that produce 99,432 three word training examples. The test set is created by first parsing Section 00 using the Bikel (2004) parser. We then extract the base-NPs that the parser identifies and insert the gold-standard NP bracketing for evaluation. We reject brackets that cross an NP boundary (i.e. a parsing error). This results in a test set of 3005 complex NPs.

We then carry out a similar analysis as above for complex NPs. In these experiments, we introduce novel features based on the parser's output. Firstly, the labels of the parent and grandparent of the NP, which have been informative in PCFG parsing, are included. Secondly, we add features for the head-word of the parent and the grandparent, and generalise these using their POS and NER tags. Finally, we add the Web 1T counts for the parent and grandparent head-words.

The results of these experiments, including subtractive analysis on the feature types, are shown in Table 8. Unfortunately, we find that many of the features are not helpful, and our best model utilises only the Web 1T, lexical, POS and non-head word features.

This is quite different to the results we encountered for complex NPs. We believe this is because the test NPs produced by the parser may be incorrect, while the model is trained on gold-standard NPs. Together with the brackets that we rejected for crossing NP boundaries, this would introduce a noticeable amount of noise, and we would expect lower performance in a number of feature types.

We now perform the additional task of labelling the brackets. There are only two labels to distinguish between (NML and JJP), and they can be inferred directly from the POS tag of the head. If it is a verb or an adjective, we label the node as JJP, and otherwise it is a NML. A small number of errors are introduced by this method, because of annotation errors in the Penn Treebank POS tags and by Vadas and Curran (2007), as well as errors in head finding.

Having completed the bracketing, we can then reinsert the complex NP structure from the post-processing module back into the parser output. Table 7 shows the results. The unlabelled F-score on base-NPs was 84.35% (cf. 83.59% labelled), demonstrating the efficacy of our simple labelling process. The post-processing module improved on

the parser's F-score (cf. Table 6) by 31.72% for base-NPs, and by 8.13% on all NML and JJP nodes. Finally, the post-processing system has increased performance on all constituents, removing some of the loss incurred in order to make NP bracketing possible.

6 Conclusion

In this paper, we have presented a comprehensive analysis of the noun phrase bracketing task, investigating the performance of many different data sets, features, and models. We started with compound noun phrases, as in Lauer (1995), and used models developed for that task to bracket longer, complex NPs with Barker's (1998) algorithm. Finally, we incorporate the complex NP model into a post-processor for the Bikel (2004) parser.

In every evaluation we carried out, our supervised model improved significantly upon the baseline unsupervised results. Although this is not surprising, it is the first time that sufficient annotated data has been available for supervised methods to be competitive on this task. We achieved 93.01% F-score on our simple NP test set, compared to an unsupervised model's 84.14%. With complex NPs, our improvement was even greater: 91.44% F-score on matched brackets, compared to 32.40%. Our supervised approach also incorporates several novel feature types extracted from within and surrounding the NP to be bracketed.

Vadas and Curran (2007) found that inter-annotator agreement for NP bracketing was 98.52% (exact NP match), so while our best result of 93.03% is competitive, there is still room for significant improvement. We intend to continue these experiments with more detailed feature analysis, and an exploration of different machine learning methods. There are also possible improvements to Barker's (1998) algorithm, e.g. incorporating the probability with which each decision is made.

Using the large amount of Penn Treebank data annotated by Vadas and Curran (2007), we have shown that complex NP bracketing can be performed with a level of accuracy necessary to be exploited by many practical applications. Our system can be used in conjunction with a widely used parser in order to identify sub-NP structure, thereby increasing performance on question answering, anaphora resolution, and many other downstream NLP tasks.

Acknowledgements

We would like to thank Preslav Nakov, for providing us with the code and data for his NP bracketing system; and the anonymous reviewers for their helpful feedback. This work has been supported by the Australian Research Council under Discovery Projects DP0453131 and DP0665973.

References

Ken Barker. 1998. A trainable bracketer for noun modifiers. In *Proceedings of the Twelfth Canadian Conference on Artificial Intelligence (LNAI 1418)*, pages 196–210. Vancouver.

Dan Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, University of Pennsylvania.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Technical report.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name`, implementation available at `http://hal3.name/megam/`.

Hal Daumé III and Daniel Marcu. 2004. Np bracketing by maximum entropy tagging and svm reranking. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 254–261. Association for Computational Linguistics, Barcelona, Spain.

Usama M. Fayyad and Keki B. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artifical Intelligence (IJCAI–93)*, pages 1022–1029. Chambery, France.

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

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Journal of Computer Speech and Language - Special Issue on Multiword Expressions*, 19(4):313–330.

Mirella Lapata and Frank Keller. 2004. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of NLP tasks. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 121–128. Boston.

Mark Lauer. 1995. Corpus statistics meet the compound noun: Some empirical results. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, MA.

Mitchell Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA.

Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Preslav Nakov and Marti Hearst. 2005a. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of CoNLL-2005, Ninth Conference on Computational Natural Language Learning*. Ann Arbor, MI.

Preslav Nakov and Marti Hearst. 2005b. A study of using search engine page hits as a proxy for n-gram frequencies. In *Proceedings of the Recent Advances in Natural Language Processing Conference*. Borovets, Bulgaria.

James Pustejovsky, Sabine Berger, and Peter Anick. 1993. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*. Cambridge MA, USA.

Philip Resnik. 1993. *Selection and Information: A Class-Based Approach to Lexical Relationships*. Ph.D. thesis, University of Pennsylvania.

Satoshi Sekine and Michael Collins. 1997. EVALB bracket scoring program. Available from `http://nlp.cs.nyu.edu/evalb/`.

David Vadas and James R. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report.

Empirical study on Parsing Chinese Based on Collins' Model

Hailong Cao, Yujie Zhang, Hitoshi Isahara
Computational Linguistics Group, National Institute of Information and Communications Technology
{hlcao, yujie, isahara}@nict.go.jp

Abstract

There is a large set of details in Collins' model which has been proved to be quite important in English parsing. When we apply the model to parse other languages such as Chinese, it is very important to know whether these details will still work. We present a careful assessment of the applicability of a number of detail features in Collins' model when adapted to Chinese. We find that most of them work well for Chinese parsing; some of them do not help or even hurt the performance of Chinese parser. By combining a lexicalized tagging model and Collins' model, we present the syntactic parsing result on the newly available CTB5.1 and achieve 84.45% precision and 84.34% recall on the standard test set.

1 Introduction

The development of Penn Chinese Treebank (abbreviated as CTB, Xue et al., 2005) spurred the research of Chinese syntactic parsing. Many English parsing models have been adapted to parse CTB. Although many improvements have been achieved during the last few years, the label precision and recall of state of art of Chinese parsing is just about 80% and 82% respectively, lower than that of English. There are two obvious reasons accounting for that. First, most previous works trained their parsing models on the first version of CTB that is too small to estimate parameters reliably. Second, the models used in most previous work were originally proposed for English and were likely to be sub-optimal for Chinese.

In this paper, we address the above two issues by presenting an empirical study of adapting Collins' model(Collins, 1997; Collins, 1999) to Chinese based on the newly available CTB5.1 that is more than four times larger than the first version of CTB. There is a large set of details in Collins' model which has been proved to be quite important in English parsing. By taking advantage of the source code released by Collins and the details documented by (Bikel, 2004(a)), we provide a thorough analysis of the effect of some features and details on Chinese parsing. We found that most of them work well for Chinese parsing; some of them do not help or even hurt the performance of Chinese parser. By combining a lexicalized tagging model and Collins' model, we present the syntactic parsing result on the CTB5.1 and achieve 84.45% precision and 84.34% recall on the standard test set.

The rest of this paper is organized as follows. In section 2, we describe the part of speech (POS) tagging model which is a necessary preprocessor of the parsing model. Section 3 briefly introduces Collins' parsing model. Section 4 gives the POS tagging results. In section 5, we analyze the effect of base noun phrase model, coordination model and distance feature, examine the influence of POS tagging errors on parsing performance. Related work on CTB parsing is reviewed in section 6. Finally, conclusion is made in section 7.

2 Lexicalized POS Tagging Model Based on HMM

Our parser takes segmented sentences as input; formally it is a sequence with n words:

$$W = w_1, w_2, \ldots\ldots w_n$$

Before parsing the sentence, we will assign each word in the sentence an appropriate part of speech tag by a lexicalized hidden Markov model (HMM).

Usually there are more than one POS tag sequences for a given word sequence since there are usually more than one POS tags for a single word. The statistical POS tagging method based

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 113–119,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

on Bayesian model is capable of assigning a POS tagging sequence with the greatest conditional probability, which is shown as follows:

$$Tag_{best} = \arg\max_{Tag} P(Tag \mid W) = \arg\max_{Tag} \frac{P(Tag, W)}{P(W)} = \arg\max_{Tag} P(Tag, W) \quad (1)$$

Where $Tag = t_1, t_2, ...t_n$ is a candidate POS sequence for *W*.

The classical HMM assumes that the transformation from one state (that means POS here) to another is not affected by the current observation value (that means the current word), and the generation of current observation value is independent from other observation values. That is:

$$P(Tag, W) = P(Tag)P(W \mid Tag) \approx \prod_{i=1}^{n} P(t_i \mid t_1, t_2 ....., t_{i-1}) \prod_{i=1}^{n} P(w_i \mid t_1, t_2 ....., t_n) \quad (2)$$

Furthermore, only N previous states are considered when the current state is generated. And only the current state is involved when the current word is generated:

$$P(Tag, W) = P(Tag)P(W \mid Tag) \approx \prod_{i=1}^{n} P(t_i \mid t_{i-N}, t_{i-N+1} .....t_{i-1}) \prod_{i=1}^{n} P(w_i \mid t_i) \quad (3)$$

This is the so-called N-order model or the (N+1)-gram model. In practice, bi-gram or tri-gram model is often used to alleviate data sparseness.

In fact, we observed that there is close association between POS tags and words in Chinese text, the above model can not reflect the characteristic of Chinese very well. In order to capture the relation between POS tags and words in Chinese text, we augment HMM by the method below:

$$Tag_{best} = \arg\max P(Tag, W) = \arg\max \prod_{i=1}^{n} P(t_i, w_i \mid t_1, w_1 .....t_{i-1}, w_{i-1}) \approx \arg\max \prod_{i=1}^{n} P(t_i, w_i \mid t_{i-1}, w_{i-1}) \quad (4)$$

By doing this transformation, we can correct the HMM's unpractical assumption introduce lexical information into POS tagging model to strengthen its discriminative ability.

However, data sparseness problem becomes serious after we introduce lexical information. So it is necessary to utilize some data smoothing method. From equation (4), we can get:

$$P(t_i, w_i \mid t_{i-1}, w_{i-1}) = P_1(t_i \mid t_{i-1}, w_{i-1}) P_2(w_i \mid t_{i-1}, w_{i-1}, t_i) \quad (5)$$

In this way, we can smooth the P_1 and P_2 in equation (5) by the following method:

$$P_1(t_i \mid t_{i-1}, w_{i-1}) \approx \lambda_1 P_{MLE1}(t_i \mid t_{i-1}, w_{i-1}) + (1 - \lambda_1) p_{MLE1}(t_i \mid t_{i-1}) \quad (6)$$

$$P_2(w_i \mid t_{i-1}, w_{i-1}, t_i) \approx \lambda_{21} P_{MLE\ 2}(w_i \mid t_{i-1}, w_{i-1}, t_i) + (1 - \lambda_{21})[\ \lambda_{22}(P_{MLE\ 2}(w_i \mid t_{i-1}, t_i) + (1 - \lambda_{22}) P_{MLE\ 2}(w_i \mid t_i)\ ] \quad (7)$$

λ_1, λ_{21} and λ_{22} are smoothing parameters and $P_{MLE}(x|y)$ is the empirical probability estimated from the data in the training set by using maximal likelihood estimation method:

$$P_{MLE}(x \mid y) \equiv \frac{count(x, y)}{count(y)} \quad (8)$$

3 Parsing based on Collins' Model 2

The parsing model we start with is the well-known head-lexicalized model proposed by Collins. Given an input sentence $S=(w_1/t_1, \ldots\ldots w_n/t_n)$ the most likely parse tree defined by a statistical generative model is:

$$T_{best} = argmaxP(T \mid S) = argmax\frac{P(T,S)}{P(S)} = argmaxP(T,S) \quad (9)$$

Probabilistic context-free grammar (PCFG) is one of the simple methods that are used to model distributions over sentence/parse-tree pairs. If there are *k* context free grammar rules in the parse tree, then

$$P(T,S) = \prod_{i=1..k} P(RHS_i \mid LHS_i) \quad (10)$$

Where LHS /RHS standards for the left/right hand side of the grammar rule.

Based on PCFG, Collins proposed a lexicalized model by associating a word *w* and a part of speech tag *t* with each non-terminal node in the parse tree. Formally, a grammar rule LHS $\rightarrow$ RHS can be written as:

$$Parent(t,w) \rightarrow L_m(t,w) \ldots\ldots L_1(t,w)\ H(t,w)\ R_1(t,w) \ldots\ldots R_n(t,w)$$

Where *Parent* is the father node and *H* is the head child, $L_m \ldots\ldots L_1$ and $R_1 \ldots\ldots R_n$ are left and right modifiers of H.

To overcome the sparseness problem caused by the addition of lexical items, the generation of RHS is broken down into a Markov process that makes certain independence assumptions, and the probability of a grammar rule is defined as:

$$P(RHS \mid LHS) = P_h(H \mid Parent(t,w)) \cdot \prod_{i=1}^{m+1} P_l(L_i(t,w) \mid Parent(t,w), H) \cdot \prod_{i=1}^{n+1} P_r(R_i(t,w) \mid Parent(t,w), H) \quad (11)$$

Where L_{m+1} and R_{n+1} are stop categories. The probability P_h, P_l and P_r are estimated by maximum likelihood estimation method.

Collins' model 2 also includes subcategorization frames. So it is necessary to make complement/adjunct distinction in training data. We label the following three types of non-terminal as complement:

(1) NP, CP (Sub clause) or IP (simple clause) whose parent is IP.
(2) NP, CP, VP or IP whose parent is VP.
(3) IP whose parent is CP.

In addition, the non-terminal will not be labeled as complement if it is the head child of its parent. For more details such as parameter estimation and special preprocessing of punctuations, we refer the reader to (Collins, 1999).

4 Experiments on tagging

4.1 Data

In our experiments, both the tagging model and the parsing model are trained and tested on the Penn Chinese Treebank 5.1 which contains 507,216 words, 18,782 sentences. Following the previous researches, we use the article 271-300 for testing and 301-325 for developing. All the other articles, i.e. article 001-270 and article 400-1151, are used for training.

The POS tagset of CTB has 33 tags, fewer than that of English Penn Treebank (abbreviated as ETB) which has 45 tags. This is consonant with the fact that Chinese make less use of morphology than English. For example, ETB uses 6 verb tags, namely VB, VBD, VBG, VBN, VBP and VBZ, to express the distinctions of tense and person. However, such distinctions are not necessary for Chinese.

The definitions of most of the CTB tags such as NN, CD and CC are similar to that of ETB. CTB also has several language special tags such as DEC, DEG and BA.

4.2 Result and Analysis

When we train the tagging model, all syntactic labels in CTB are removed and only words and POS tags are kept.

Table1: Tagging results

Model	Accuracy on development set	Accuracy on test set
Bi-gram HMM	91.29%	91.95%
Lexicalized model	93.27%	94.65%

For comparison, we use the bi-gram HMM as a baseline for the lexicalized tagging model. Table 1 shows the evaluation results. From table 1, we can see that the performance of lexicalized model outperforms bi-gram HMM significantly. Table 2 gives the top 10 frequent error types.

Table 2: High frequency tagging error types on development set

Error type		Count	Percentage
Gold tag	Error tag		
NN	M	11	2.40%
VA	NN	13	2.83%
JJ	NN	16	3.49%
CD	NN	20	4.36%
DEG	DEC	20	4.36%
NR	VV	20	4.36%
DEC	DEG	30	6.54%
NN	VV	31	6.75%
VV	NN	40	8.71%
NR	NN	156	33.99%

The Gold tag and Error tag in table 2 is used to express that Gold tag is mistagged as Error tag. The most frequent error occurs on NR(name entity) which is mistagged as NN(noun) for 156 times. NR words are difficult to recognize because many of them are never seen in the training data. How to deal with unknown words is a direction of future work.

NN/VV ambiguity is widespread in Chinese and is very difficult to resolve due to the lack of morphology in Chinese.

5 Experiments on parsing

5.1 Setting

Before we train the parsing model, we also do the standard tree transformation such as the removal of empty nodes and semantic information in the tree bank. The head percolation table from (Xia, 1999) is used to find the head of constituent in CTB. For convenience, all the following experiments are implemented on sentences which have no more than 40 words.

CYK parsing algorithm is used to decode the model in a bottom-up process. Beam search is used to prune search space. We set the beam width to 50,000. We use colon, comma and quotation mark as clues for comma-pruning.

From a more general view, there is another pruning strategy. Collins set a limitation for the maximum number of edges in the chart table. For a sentence to be parsed, if the number of edges generated by the decoder is more than two third of the limitation, the parsing algorithm will exit. This limitation was originally set to 200 thousand. However, if we still use this limitation in Chinese parsing, 10 out of 348 sentences from test set will be failed to get any result because of too much parse ambiguity in Chinese. So we raise the limitation from 200 thousand to 2 million to guarantee that the decoder can find a parsing tree for every sentence in the development set and test set.

5.2 The effect of BNP & coordination model

There are two important details that Collins used in his English parser.

1) One is the special processing of base noun phrase (BNP), i.e. the non-recursive noun phrase. Because the annotation of Penn Treebank is different from what Collins model has assumed, Collins re-annotated the non-recursive noun phrase as BNP and inserted an additional noun phrase node above the BNP. For example, the parse tree shown in figure 1 will be transformed into the style illustrated in figure 2. The English translation of the example sentence would be: "In Ning-bo(宁波) bonded-area(保税区), the construction(建设) achievement(成就) is remarkable(显著) ".

2) The other detail is the coordination model. There is a coordinator in the coordination construction. The model described in section 3 fails to learn that there is always one phrase following the coordinator. For this reason, instead of generating the coordinator and the following phrase one by one independently, they are generated together in one step.

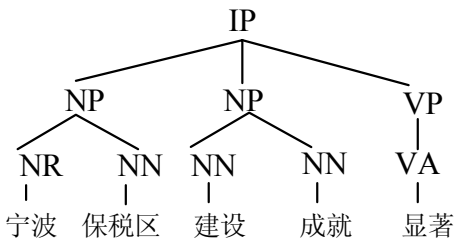


Figure 1: A sample parse tree from CTB.

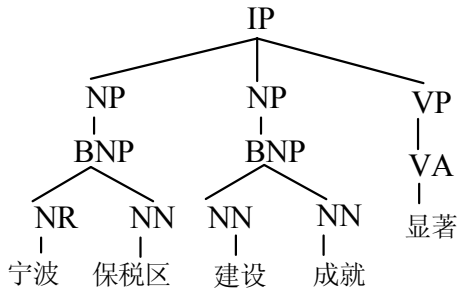


Figure 2: A sample parse tree after re-annotation.

BNP model and coordination model are originally proposed to deal with the language phenomena of English and the annotation standard in the Penn English tree bank. So it is still an open question whether the special details are also effective on Chinese which is different from English in many aspects. In order to test that, we performed four experiments on the development set. Table 3 shows the results.

Table 3: Results on development set ("Yes" means the detail is used, and "No" means it is not used.)

BNP	Coordination	Precision	Recall	F1
No	No	81.66%	81.21%	81.43%
Yes	No	82.57%	82.48%	82.53%
No	Yes	81.90%	81.27%	81.58%
Yes	Yes	82.34%	82.37%	82.36%

It is clear that BNP model can make significant improvement. It raises the F1 from 81.43% to 82.53%. Coordination model can also make a little improvement. However, if BNP model and coordination model are utilized together, the performance is worse than that when we only use BNP model.

We think one possible reason is the head modification in coordination phrase. In Collins' model, the head of coordination is determined by two steps. First, it is picked out by the head percolation table. Second, if the head is on the right of the conjunction which is non-initial, the non-

terminal on the left of the conjunction will be chosen as the new head. For example, in a grammar rule such as $NP \rightarrow NP_1$ CC NP_2, NP2 is the initial head child of NP. But when training the coordination model, the head is modified to be NP_1. However, according to the head percolation table, Chinese noun phrase and verb phrase are usually right-headed. So the head of coordination and non-coordination is not determined in a consistent way. We suspect this is the reason why coordination model hurts the parsing performance. So rather than follow the original model, we always select the non-terminal on the right of CC as the head both on training and decoding process. The experiment results are shown in table 4.

Table 4: Results on development set with new coordination model.

BNP	Coordination	Precision	Recall	F1
No	Yes	82.07%	81.81%	81.94%
Yes	Yes	82.23%	82.39%	82.31%

In this way, we can raise the F1 from 81.58% to 81.94% when only coordination model is used. But coordination model still hurts performance when being used together with BNP model. We leave the analysis to the future work and tentatively conclude that the coordination model is not necessary when we build Chinese parser. In all the following experiments, BNP model is kept and coordination model is eliminated.

5.3 The Impact of the Distance Measure on Accuracy

It has been proved that distance measure is very important in English parsing. There are two features in the distance measure. One is the adjacency condition which indicates whether the modifier is adjacent to the head or not. The other is verb condition which indicates whether there is a verb between the modifier and the head or not. For more strict definition of distance, we refer the reader to (Collins, 1999). One important motivation of adding distance measure into the model is to capture the right-branching structures in English. However, Chinese is not a right-branching language. So it is necessary to check the impact of distance measure on Chinese parsing. Table 5 gives the results with and without the adjacency and verb distance measure.

Table 5: Results on development set with and without distance measure.

A	V	Precision	Recall	F1
No	No	78.36%	75.59%	76.95%
Yes	No	82.06%	82.17%	82.12%
Yes	Yes	82.57%	82.48%	82.53%

It is clear that adjacency condition is very effective. It raises the F1 from 76.95% to 82.12%. However, verb condition only adds 0.41% improvement. What we found in table 5 is quite different from that in English parsing. For a comparison, we list the effect of distance measure on English parsing in table 6.

Table 6: Results on section 0 of WSJ Treebank. (From (Collins, 1999))

A	V	Precision	Recall
No	No	85.1%	86.8%
Yes	No	87.7%	87.8%
Yes	Yes	88.7%	89.0%

We can get two possible clues to improve Chinese parsing by comparing the results.

First, verb condition works much more effective at English parsing than at Chinese parsing. We think the current definition of verb condition is sub-optimal for Chinese and it is quite possible to improve Chinese parsing by further tuning it.

Second, there is a more dramatic performance drop on Chinese than English when we turn off the distance measure. This difference deserves our attention because the model still includes subcategorization frame which has been shown to have similar effect with distance measure at English. So the difference implies that we have not made full use of subcategorization in Chinese parsing. In fact, the complement/adjunct distinction we make in section 3 is very simple and it is quite possible to improve Chinese parsing by further tuning it.

5.4 The influence of POS tagging errors on parsing

Because of the shortness of training data, the accuracy of our tagger is rather lower than the state-of-art English POS tagging. In order to know how much parsing errors is caused by POS tagging error, we rerun the parser on the sentences with gold standard POS tags.

In this way, we can raise the F1 from 82.53% to 86.37%. However, this result does not tell the whole story. In fact, tagging is integrated in Collins' model and every possible POS tag is considered if the word has been seen in training

data. As for the output of POS tagger, the parser only accepts the POS tags of unseen words and tags other words by itself. So in the above experiment, there are still many parsing errors due to the parser fails to select the correct POS tags for some words.

To eliminate the influence of tagging errors, we change the setting of Collins' model to make sure that only the gold standard POS tag is considered. Then we have a parsing performance of 89.10%. This result is very promising. If more training data is available, we think there is large room to improve the tagging performance and therefore the parsing performance.

Finally, we run the parser on the automatically tagged sentences from the standard test set. We achieve 84.45% precision and 84.34% recall.

6 Related Work on Parsing CTB

Much work has been done on parsing CTB. Table 7 gives some previous results.

Table 7: Results from related work on the standard test set.

	Precision	Recall	F1
Bikel & Chiang 2000	77.8%	76.8%	77.3%
Chiang & Bikel 2002	81.1%	78.8%	79.9%
Levy & Manning 2003	78.4%	79.2%	78.8%
Hearne & Way, 2004	77.92%	74.46%	76.15%
Jiang's Thesis 2004	82.0%	80.1%	81.1%
Bikel's Thesis 2004(b)	81.2%	78.0%	79.6%
Xiong et al. 2005	80.1%	78.7%	79.4%
Wang et al. 2006	81.1%	79.2%	80.1%

(Bikel and Chang, 2000) presented the first result of CTB parsing based on BBN model and TIG model. (Chang and Bike, 2002) proposed an automatic method to determine the head child based on the EM algorithm. (Levy and Manning, 2003) adapted factored model to Chinese parsing and achieved much improvement by grammar transformation inspired by error analysis. (Hearne and Way, 2004) applied Data-Oriented Parsing approach to CTB. (Jiang, 2004) adapted Collins' model to Chinese parsing and made significant improvement by proposing language specific enhancements. (Bike, 2004(b)) built a multi-language parsing engine which can be extended to Chinese by the addition of tweaked head-finding rules and argument-finding heuristics. (Xiong et al., 2005) proposed a semantic-class based method by combining factored model and Collins' model. They used thesaurus to overcome the data sparseness problem. (Wang et al., 2006) presented a deterministic Chinese parser. They transform parsing into a succession of classification problem and make it convenient to apply machine learning method. In addition, (Luo, 2003) and (Fung et al., 2004) constructed character based parser. So their work is not directly comparable with the other parsers that operate at word-level.

We note that Collins' model have already been applied to parse Chinese in several work. However, given the same training data and test data, the obtained results are different from each other greatly. We think the different way of using the large sets of details accounts for most of the difference. Different POS taggers used in each work also result in different parsing accuracy to some extent.

It should be noticed that the recall is much lower than the precision in most of results shown in table in 7. We think one possible reason for that is the shortness of data since all of those parsers are trained on the first version of CTB that consists of only 4185 sentences.

7 Conclusion

There is a large set of details in Collins' model which has been proved to be quite important in English parsing. We present a thorough analysis of the effect of some important details on Chinese parsing. We find that BNP model works well for Chinese parsing while coordination model does not help or even hurts performance of Chinese parser. As for the effect of distance feature, we find that the adjacency condition is very effective in Chinese parsing while the effect of verb condition on Chinese is not as great as that on English. We also evaluate the influence of POS tagging errors on parsing and find much parsing errors are aroused by POS tagging errors. By combining a lexicalized tagging model and Collins' model, we present the syntactic parsing result on the newly available CTB5.1 and achieve 84.45% precision and 84.34% recall on standard test set.

The current model is deficient in analyzing coordination structure. In future work, more effective model should be designed.

References

Daniel M. Bikel and David Chiang. 2000. Two Statistical Parsing Models Applied to Chinese Treebank. In *Proceedings of the 2nd Chinese language processing workshop.*

Daniel M. Bikel. 2004(a). Intricacies of Collins' Parsing Model. *In Computational Linguistics*, 30(4): 479-511.

Daniel M. Bikel. 2004(b). *On the Parameter Space of Generative Lexicalized Statistical Parsing Models.* Ph.D. thesis, University of Pennsylvania.

David Chiang and Daniel Bikel. 2002. Recovering Latent Information in Treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics.*

Deyi Xiong, Shuanglong Li, Qun Liu et al. 2005. Parsing the Penn Chinese Treebank with Semantic Knowledge. In *Proceedings of the Second International Joint Conference Natural language processing.*

Fei Xia. 1999. *Automatic Grammar Generation from Two Different Perspectives.* PhD thesis, University of Pennsylvania.

Mary Hearne and Andy Way. 2004. Data-Oriented Parsing and the Penn Chinese Treebank. In *Proceedings of the First International Joint Conference Natural language processing.*

Mengqiu Wang, Kenji Sagae and Teruko Mitamura. 2006. A Fast, Accurate Deterministic Parser for Chinese. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics.*

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Conference of the Association for Computational Linguistics.*

Michael Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing.* Ph.D. thesis, University of Pennsylvania.

Nianwen Xue, Fei Xia, Fu-Dong Chiou and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11(2): 207-238.

Pascale Fung, Grace Ngai, Yongsheng Yang and Benfeng Chen. 2004. A Maximum-Entropy Chinese Parser Augmented by Transformation-Based. Learning. *ACM Transactions on Asian Language Processing*, 3(2):159-168.

Roger Levy and C. Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? In *Proceedings of the 41st Conference of the Association for Computational Linguistics.*

Xiaoqiang Luo. 2003. A Maximum Entropy Chinese Character-Based Parser. In *Proceedings of the conference on Empirical methods in Natural Language Processing.*

Zhengping Jiang. 2004. *Statistical Chinese parsing.* Honours thesis, National University of Singapore.

Learning-Based Argument Structure Analysis of Event-Nouns in Japanese

Mamoru Komachi, Ryu Iida, Kentaro Inui and Yuji Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
Nara, 630-0192, Japan
{mamoru-k,ryu-i,inui,matsu}@is.naist.jp

Abstract

Previous work on predicate-argument structure analysis has focused largely on verbs and adjectives. However, to extract richer information from such a sentence as "His decision was right," not only the predicate "was right" but also the noun "decision" need to be identified as an event denoting expression. Here, we call event denoting nouns *event-nouns*, which refer to events like "he decided something." We define a task of argument structure analysis of event-nouns in Japanese. Some event-nouns, such as *ryouri* "cook/cuisine", do not always refer to an event, and word sense disambiguation of event-nouns has to be done depending on their context. In this paper, we present an unsupervised machine learning method to extract patterns of event-nouns for event-hood determination. We also investigate verb-noun co-occurrences and support verb constructions for argument identification of event-nouns.

1 Introduction

The beginning of predicate-argument structure analysis using machine learning methods goes back to Gildea and Jurafsky (2002). They defined the semantic role labeling(SRL) task as finding core and adjunct arguments of predicates or verbs in the FrameNet's (Fillmore and Baker, 2000) semantics. The PropBank (Palmer et al., 2005) is one of the corpora annotated with semantic roles, which adds semantic analysis to the parsed syntactic tree of the Penn Treebank (Marcus et al., 1993) based on the VerbNet (Kipper et al., 2000). Using the PropBank, evaluation campaigns were carried out at the CoNLL 2004 and 2005 shared tasks [1], and there have been a number of improvements in machine learning methods with respect to SRL.

In addition to the predicate-argument structure, there have been several projects annotating the sets of arguments that co-occur with nouns. NomBank (Meyers et al., 2004b) is one of the projects that annotate nouns and their arguments. It focuses on nominalization of verbs in English, and thus it follows the PropBank scheme to add a semantic information layer to the Penn Treebank. For example, in a sentence like "His decision was to leave." one can assume that the noun "decision" refers to an event [REL=*decision*, A0=*his*, A1= *to leave*]. We call such a kind of nouns *event-nouns* and semantic roles like X and Y *arguments*. Despite the fact that the predicate-argument structure analysis is actively studied, there are only a few analysis models developed for nominalized predicates (Jiang and Ng, 2006; Xue, 2006).

In this paper, we discuss several issues in the argument structure analysis of event-nouns in Japanese. The main contributions of this work are (1) the task definition and corpus annotation of event-nouns, (2) using unsupervised method to extract syntactic contexts from large corpora, and (3) building a co-occurrence model and a resource of support verb constructions for the argument structure analysis of event-nouns. The remaining sections of this paper are organized as follows: we first describe the argument structure analysis of event-nouns in section 2. We divide argument structure analysis of event-nouns into two sub-tasks: event-hood determination and argument identification, and propose an unsupervised method to learn contextual clues for determination in section 3. We then propose a supervised method using verb-noun co-occurrence statistics

[1] http://www.lsi.upc.edu/~srlconll/

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 120–128,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

and syntactic clues for argument identification in section 4. Finally, we conclude our work and present future directions in section 5.

2 Argument Structure Analysis of Event-Nouns in Japanese

Argument identification of event-nouns is similar to the semantic role labeling (SRL) task for predicates. However, compared to argument structure analysis of predicates, there is a major difference between the argument structure analysis of predicates and event-nouns.

The problem is that some event-nouns have several meanings including an event-reading and other readings. This is especially important in Japanese because verbal nouns, which refer to the same type of events as the base verbs denote, have the same surface form as their base verbs. For example, given a sentence

(1) *watashi-wa* *saikin* *koushu-denwa*$_i$*-kara*
I-NOM recently public phone-ABL[2]
kanojo-ni *denwa*$_j$*-o* *shi-naku-na-tta*
her-DAT call-ACC do-NEG-become-PAST
Recently, I stopped using public phones to call my girl friend.

The common noun *denwa*$_i$ refers to a phone as a physical object while the verbal noun *denwa*$_j$ refers to an event of someone's calling somebody else. Note that both nouns denwa$_i$ and denwa$_j$ have the same surface form, and the verbal noun *denwa*$_j$ has the base verb *denwa-suru* "make a call". Thus, word sense disambiguation has to be done if a noun can denote both an object and an event. For that purpose, we define *event-hood*[3] as whether a given verbal noun refers to an event or an object. If there is a noun that refers to an event, these noun is said to have event-hood.

To address this issue, we divide the argument structure analysis of event-nouns into two major tasks. The first task is *event-hood determination* and the second task is *argument identification*. Event-hood determination is the task of determining whether event-nouns refer to events or not given in a context. Argument identification is the task of identifying the arguments of event-nouns which refer to events.

The concept of event-nouns is a superset of result nominals, *simple* event nominals, and *complex* event nominals described in Grimshaw (1990). The difference between *simple* and *complex* event nominals is that *complex* event nominals have an associated event structure and argument structure. Since verbal nouns are a subset of *complex* event nominals, we can exploit semantically rich lexicons of verbs for the argument identification task. [4] There are a number of simple event nominals like 'congress', which do not have a direct correspondence to verbs, but they are beyond the scope of this paper.

2.1 Description of Annotation

One interesting issue in the annotation of the argument structure of event nouns is how we abstract the argument structure. The PropBank and NomBank use semantic roles as a label to the arguments structure. However, it is still disputable whether or not to use semantic roles to label arguments at least in Japanese. We chose cases instead of semantic roles to annotate Japanese texts for the following reasons:

1. there is still only limited consensus on the semantic granularity and the set of semantic relations we need in Japanese,
2. the mapping rule from cases to semantic roles in Japanese seems straightforward if one has access to semantically rich resources like the VerbNet, and
3. manual annotations of semantic roles tend to be more expensive than those of cases, such as nominative, accusative and dative.

Therefore, we annotate the argument structure of event-nouns in terms of cases. We use surface cases for each event as if it were expressed in a single sentence using the active voice. For instance, in the sentence (1), we annotate [REL=*denwa*$_j$, NOM=*watashi*, DAT=*kanojo*].

Also, when annotating Japanese texts, zero-anaphora (intra- and inter-sentential zero pronouns) and exophoric relations must be taken care of. We annotate all inter-sentential, intra-sentential and exophoric relations for each event-noun. Obligatory cases vary from event-noun to event-noun, and we chose nominative, accusative and dative for a starting point. We do not annotate other cases at the moment because they are less reliable to annotate than these three cases.

[2] The instrumental case is not obligatory for denwa$_j$

[3] Event-hood here does not discriminate between a specific instance and the generic sense of an event, and we do not differentiate them.

[4] The method described in this paper would also apply to the languages other than Japanese which have nominalization.

We asked two annotators to annotate the same portion of the Kyoto Text Corpus, and calculated inter-annotator agreement of the corpus for 30 randomly selected sentences. The annotation results are evaluated by calculating recall and precision in which one annotation result is regarded as target instances and the other as selected instances. Note that arguments of event-nouns are considered for calculation of recall and precision only when event-hood are annotated by both annotators. The results are shown in Table 1.

Table 1: Inter-annotator agreement of event-nouns

	precision		recall	
event-hood	0.905	(1281/1415)	0.810	(1281/1582)
nominative	0.798	(1038/1300)	0.804	(1038/1291)
accusative	0.893	(469/525)	0.765	(469/613)
dative	0.717	(66/92)	0.606	(66/109)

With regard to inter-annotator agreement, event-hood annotation and nominative/accusative case annotation achieve high results, although the annotation of the dative case shows relatively poor inter-annotator agreement. The reason seems to be that the dative case is less obligatory than the other two cases and the two annotators do not agree on the annotation criteria. We plan to build and use a case frame dictionary based on verbs to minimize mis-classification of case frames.

The distribution of each case (nominative/accusative/dative) in our corpus is shown in Table 2. Exophora is a reference in a text or utterance to something external to it, which is only fully intelligible in terms of information about the extralinguistic situation.

From Table 2, we can see that nominative cases distribute widely, whereas a large number of accusative and dative cases exist in the same chunk or in dependency relations.

Our event-tagged corpus (Iida et al., 2007) is available online [5] for research purposes and has been downloaded by a wide variety of affiliations.

2.2 Related work

For the task of argument structure analysis of event-nouns for Japanese, the Kyoto Text Corpus Version 4.0 (Kurohashi and Nagao, 1997) has been annotated with the argument structure of event-nouns. However, they do not explicitly discuss the task of event-hood determination and there is no discussion on agreement ratio of the corpus. We annotated the whole of the Kyoto Text Corpus Version 3.0, whereas only 12% of the Kyoto Text Corpus Version 4.0 was annotated with cases.

For the analysis model, Jiang and Ng (2006) proposed a maximum entropy approach to the argument identification of event-nouns. They explored several features specific to event-nouns. For example, they added features indicating whether the argument depends on the event-noun, whether the event-noun depends on a predicate, and so forth to let a machine learning classifier capture support verb constructions. However, they neither investigated features such as co-occurrence for identifying global arguments which are syntactically distant from the event-nouns, nor used a lexicon to identify support verb constructions.

3 Event-hood Determination of Nouns

3.1 Unsupervised Learning of Syntactic Contextual Clues

According to the Distributional Hypothesis (Harris, 1954), it can be expected that the distribution of the contexts where verbal nouns with an event-reading occur tends to be different from that of verbal nouns with an object-reading. Furthermore, the former distribution is expected to be close to the context distribution of verbal nouns that have unambiguously event-readings while the latter distribution should be close to that of common nouns with only object-readings. Based on this intuition, we propose to learn syntactic context clues useful for event-hood determination in an unsupervised fashion as follows.

To learn the patterns of noun phrases, we encoded sequences of parts of speech of morphemes (window size 3) and the sequence of morphemes in dependency relations into a tree structure, and then extract a pattern as a sub-tree of the encoded tree. POS tagging and dependency parsing were performed by ChaSen [6] and CaboCha [7].

Positive instances are unambiguous event-nouns which only have event-reading like *shimpai* "concern". We used nouns which are members of the category "Noun-Abstract-Event-hood-{Action,Event}" in Japanese Lexicon (Ikehara et al., 1997). Also, negative instances are unambiguous nouns which only have object-readings. We then defined them as proper nouns and com-

[5] http://cl.naist.jp/nldata/corpus/

[6] http://chasen-legacy.sourceforge.jp/

[7] http://chasen.org/~taku/software/cabocha/

Table 2: The distribution of nominative, accusative and dative cases of event-nouns and predicates (predicates are shown in parenthesis)

	intra-sentential			inter-sentential	exophoric
	same chunk	dependency rel.	other		
nominative	2024; 7%(0%)	6428; 23%(42%)	7131; 25%(31%)	5185; 18%(12%)	7501; 27%(15%)
accusative	5519; 50%(0%)	3424; 31%(84%)	1102; 10%(13%)	854; 8%(3%)	41; 0%(0%)
dative	841; 43%(3%)	418; 22%(88%)	470; 24%(7%)	201; 10%(3%)	10; 0%(0%)

mon nouns like *isu* "chair" in the category "Noun-Object" in the thesaurus. We used articles from one month of Mainichi Newspapers (positive instances:117,581, negative instances:282,419).

In order to capture the patterns of nouns, we used a tree classifier called BACT [8] (Kudo and Matsumoto, 2004). BACT iteratively selects weighted sub-trees effective for classifying training data by learning tree structure using the Boosting algorithm (Freund and Schapire, 1996). If it is given sentences parsed in word dependency structure, it learns syntactic patterns as rules for classifying them.

If the weighted sum of all the tree patterns occurring in a test sentence is positive BACT classifies it as positive, while the weighted sum is negative BACT classifies the instance as negative. Therefore, acquired sub-tree patterns with high positive weight are strong indication of having event-hood and those with high negative weight are strong indication of not having event-hood. Acquired sub-trees with high weights are shown in Table 3.

3.2 Experiments

For the event-hood determination task, we used Support Vector Machines [9] (Vapnik, 1998) to learn the contexts of event-nouns, and ran 10-fold cross validation to evaluate the performance of our system.

The baseline system chooses a predominant sense in our event-tagged corpus. Our proposed model takes into account the patterns of event-nouns obtained by BACT in addition to some heuristic features. Extracted patterns and the features used in the experiment are summarized in Table 5. Evaluation was performed on 80 articles (800 sentences) from newspapers. The evaluation set contains 1,659 event-nouns (575 instances have event-hood).

[8] http://chasen.org/~taku/software/bact/

[9] http://www.chasen.org/~taku/Software/TinySVM/

Table 4: Results of event-hood determination

	precision	recall	f-measure
Predominant(baseline)	60.4	88.2	71.7
Unsupervised(proposed)	73.3	80.2	76.6

Table 5: Features used for event-hood determination

Baseline features
WordForm
HeadWordFrom
HeadPOS
SemanticCategory [10]
SelectionPreference [11]
AnoB (if head word of noun A in the expression of "*A no B*")
SemanticCategoriesInChunk
POSesOfPreviousChunk
POSesOfNextChunk
POSesOfCurrentChunk
Features obtained by BACT
FollowsVerbalNoun
FollowsGeneralNoun+GeneralNoun
PrecedesVerbalNoun
PrecedesVerbalNoun+Particle
PrecedesVerbalNoun+GeneralNoun
NumOfVerbalNouns

Table 3: Examples of sub-trees effective for event-hood determination

Sub-trees with positive weights for event-hood determination	weight
(chunk-pos (verbal-noun verbal-noun))	1.13
(follows (particle verbal-noun))	1.01
(follows (noun-suffix))	0.61
Sub-trees with negative weights for event-hood determination	**weight**
(precedes (verbal-noun))	-0.92
(follows (verbal-noun))	-0.54
((follows (particle)) (precedes (verbal-noun)))	-0.50

3.3 Discussions

As shown in Table 4, sub-tree patterns improve performance of event-hood determination. Using local information to extract sub-tree patterns outperforms the baseline in precision. We would like to emphasize that our proposed method achieves at least as good as the baseline system with only a small fraction of the data using unsupervised pattern mining method from an untagged corpus, while the baseline system needs a fair amount of sense-tagged corpus. From this point of view, we intend to make further investigation on integrating compound noun and word composition information into sub-trees.

One possible way to improve event-hood determination is to use more contextual clues. For example, consider the following sentence:

(2) *tsuma-ga* *genchi-to* *denwa-no*
wife-NOM on site-DAT phone
yaritori-o *tsuduke-ta*
exchange-ACC continue-PAST
My wife continued calling the person on site.

In sentence (2), the event-noun *denwa* "phone-call" has event-hood with its nominative *tsuma* "wife". For event-hood determination, the appearance of such a noun that well satisfies the selectional preference of an argument of a verbal noun may be a good indication that the verbal noun has event-hood.

4 Argument Identification

4.1 Background

The second task of argument structure analysis of event-nouns is argument identification. Argument identification is the task of finding the arguments of an event denoted by an event-noun.

As shown in Table 2, while 84% of the accusative cases and 88% of the dative cases of predicates appear at the position which has a direct dependency relation with the corresponding predicate, only 31% of the accusative cases and 22% of the dative cases of event-nouns are in a direct dependency relation. Even when there are dependency relations between event-nouns and arguments, the arguments are not marked by case markers such as accusative case marker '*o*' and dative case marker '*ni*', while predicates usually have arguments marked by case markers.

From our corpus study, about half of the accusative and dative cases occur in the same chunk as event-nouns and form compound nouns. To deal with this issue, we tried to utilize verb-noun co-occurrences for the argument identification task of event-nouns.

The remaining 21.7% of the event-nouns are in support verb constructions. We build a case-aligned dictionary between event-nouns and predicates for identifying support verb constructions.

4.2 Using Verb-noun Co-occurrence

To confirm that we can use the verb-noun co-occurrence model for the argument identification task, we explored the verb-noun co-occurrence model using only verbal nouns which have verbs of the same surface form. An advantage of using the verb-noun co-occurrence is that we can use an automatic parser on a large amount of unlabeled data and obtain verb-noun co-occurrences marked with cases. We assume that the verb-noun co-occurrence is similar to that of event-nouns and their argument nouns.

To resolve the data-sparseness, we used a PLSI (Hoffman, 1999)-based model to smooth the verb-noun co-occurrence distribution. Fujita et al. (2004) interpret the document d in PLSI as a pair

[10] We used Bunrui Goi Hyou (National Language Research Institute, 1964) for finding semantic category of given nouns.

[11] We used the verb lexicon from EDR (National Insitute of Communication Technology, 2002) to obtain selection preferences between event-nouns and their arguments.

of $\langle v, c\rangle$ and the word w in PLSI as an n, where v, c, n stand for verb, case, and noun, respectively. The probability of verb-noun co-occurrence can be defined as follows using a latent semantic variable z:

$$P(\langle v, c, n\rangle) = \sum_{z \in Z} P(\langle v, c\rangle|z)P(n|z)P(z)$$

Z is a set of random variables denoting latent semantic classes, and PLSI reduces the dimension of the word matrix into $|Z|$ using the probability distribution. $P(\langle v, c\rangle|z), P(n|z), \text{and} P(z)$ can be estimated by EM algorithm. With the help of PLSI, one can calculate the probability of $P(\langle v, c, n\rangle), P(n|z), \text{and } P(z)$.

We used point-wise mutual information (Hindle, 1990) for co-occurrence measurement.

$$PMI(\langle v, c\rangle, n) = \log \frac{P(\langle v, c, n\rangle)}{P(\langle v, c\rangle)P(n)}$$

4.3 Building a Case Alignment Dictionary

Meyers et al. (2004a) indicate that a large number of nominalized verbs are in the support verb construction [12] and share arguments with main verbs. Jiang and Ng (2006) also address this problem, and they invented several features for a machine learning-based classifier to capture support verb construction. In contrast to their approach, since the support verb construction is a closed set and a list of support verbs can be compiled manually, we deal with this issue by developing a dictionary containing alignments from arguments of event-nouns to arguments of predicates that share arguments with event-nouns. For example, given a sentence

(3) *kare-ga* *kanojo-ni* *benkyo-wo* *oshie-ta*
he-NOM her-DAT study-ACC teach-PAST
He taught a lesson to her.

the predicate-argument structure is [REL=*oshie*, NOM=*kare*, DAT=*kanojo*, ACC=*benkyo*] and the argument structure of the event-noun is [REL=*benkyo*, NOM=*kanojo*]. In this case, the predicate *oshie* "teach" and the event-noun *benkyo* "study" share the argument *kanojo* "her" for dative and nominative, respectively.

The event-tagged corpus contains 2,173 types and 8,190 tokens of support verb constructions. Top ten frequent support verb constructions are shown in Table 6.

[12] We followed their definition of a support verb as a verb which takes at least two arguments NP_1 and XP_2 such that XP_2 is an argument of the head of NP_1.

Figure 1: Number of types and tokens of support verb constructions.

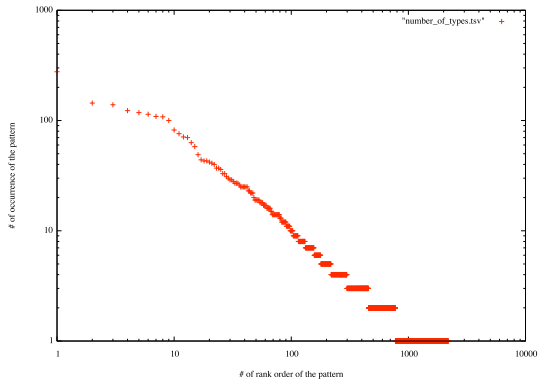


Table 6: Top ten frequent support verb constructions. S, N and E stand for a shared argument, non-shared noun phrase and event-noun, respectively.

pattern of support verb constructions	frequency
S-*ga* E-*o suru* "do"	1181(14.4%)
S-*ga* E-*o okona-u* "perform"	277(3.4%)
S-*ga* N-*o suru*	144(1.9%)
S-*ga* E-*ni suru*	139(1.7%)
S-*ga suru*	123(1.5%)
S-*ga* E-*o* N-*ni suru*	118(1.4%)
S-*ga* E-*o uke-ru* "get"	114(1.4%)
S-*ga* E-*o tsuduk-eru* "continue"	109(1.3%)
S-*ga* E-*o susum-eru* "promote"	108(1.3%)
S-*ga* E-*o meza-su* "aim to"	100(1.2%)

We extracted from a web corpus (Kawahara and Kurohashi, 2006) two million patterns of predicates on which event-nouns depend directly.[13] We then looked the top two thousands frequent pairs (it covers 80% of the entire patterns by token). We asked two annotators to test for each case given a predicate pattern whether event-nouns share any arguments with the predicate, provided by the example phrases taken from the web corpus.

Sample entry looks like [(ACC_{event}, *oshie-ru* "teach") = $\text{DAT}_{pred} \rightarrow \text{NOM}_{event}$].

Although the coverage of the dictionary of our entire event-tagged corpus (different from the web corpus) is still 49%, using the created dictionary alone for argument identification task achieves 72% in precision and 35% in recall, where precision = correct/(correct + found in the dictionary but incorrect) and recall = correct/all pred. in SVCs.

[13] Automatic dependency parsing was performed by KNP (http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp.html).

4.4 Argument Identification using Co-occurrence Statistics and Syntactic Clues

We used our event-tagged corpus for evaluation. Training and test sets were taken from two days of newspapers annotated by two annotators. We discarded tags on which two annotators did not agree. The training set consists of a day of newspapers (137 articles) and the test set consists of another day of newspapers (150 articles). We used instances that have arguments in a sentence only. The system should perform argument identification given an event-noun and its correct case frame. [14]

We used Support Vector Machines for the classifier. The implementation we used was TinySVM. We used polynomial kernel function with second degree (d = 2). Other parameters were set to the default values.

For the baseline model, we used features proposed by the Japanese zero-anaphora resolution model in Iida et al. (2006), because the task of argument identification of event-nouns is analogous to the zero-anaphora resolution task in such a way that both tasks include determination of semantic appropriateness (anaphoricity and event-hood) and identification of the best candidate. Features which were not applicable to event-nouns were eliminated.[15] Since argument identification is performed on morphological units, we added positional features within a chunk and grammatical features of POS of morphemes in a chunk.

To model the support verb construction explicitly, we added a binary feature to test whether event-nouns depend on support verbs listed in Muraki (1990). [16] Also, we added a binary feature whether the pattern of event-nouns and predicates are in the case aligned dictionary. All the novel features for argument structure analysis of event-nouns are indicated by the asterisk mark '*'.

[14] In a real application the system should also determine case frames, but we are not solving this issue for the time being.

[15] Because event-nouns syntactically differ from verbs we cannot set voice and adverb features. Also, we did not use heuristic features based on centering theory.

[16] 128 expressions are taken from Muraki (1990)

Table 7: Features used for the argument identification task

Lexical features
WordForm
Grammatical features
POS
POSesInChunk*
HasDefiniteExp
HasDemonstrativeExp
Particle
Semantic features
CoocScoreOfNewsModel
CoocScoreOfWebModel
DifferenceOfCoocScoresOfNewsModel
DifferenceOfCoocScoresOfWebModel
SelectionPreference (Ikehara et al., 1997)
PronounType
NamedEntityType
Positional features
DistanceBetweenNPAndPred
IsBegginingChunkOfSentence
IsEndingChunkOfSentence
IsBeginningMorphemeOfChunk*
IsEndingMorphemeOfChunk*
PredPrecedesNP
NPPrecedesPred
NPDirectlyDependsOnPred
NPDependsOnPred*
PredDirectlyDependsonNP
PredDependsOnNP*
PredDependsOnSupportVerb*
NPDependsOnSupportVerb*
PredSharesArguments*
InQuotation

We trained the tournament model (Iida et al., 2005) on a tuple of the event-noun, correct argument, and a candidate of an argument. For each pair, it learns features and preference which candidate wins. In decoding, it picks a candidate from a set of noun phrases one by one, and identifies as an argument the one that wins the knock-out tournament.

We compared three models with the baseline. +COOC model uses the scores of co-occurrence models as features of SVM, and +SVC model uses the binary features of the support verb constructions. We calculated accuracies for each case and model.

The accuracies of three models are shown in Table 8.

4.5 Discussions

Incorporating co-occurrence model outperforms accuracies over baseline in every case. Furthermore, using the case alignment dictionary and a

Table 8: Results of the argument identification task

Feature	NOM	ACC	DAT
Baseline	60.5	79.7	73.0
+SVC	64.2	78.0	71.4
+COOC	67.1	80.1	74.6
+SVC+COOC	68.3	80.1	74.6

list of support verbs slightly improves accuracies of nominative and dative. One of the reasons why the dictionary does not have any effect is that 90% of the accusative cases are already either in the same chunk as the event-nouns or in dependency relations. 92% of the shared arguments are nominative cases, and it is not surprising that the information of support verb construction improves the accuracy of nominative case more than other two cases.

A typical error in the deterministic argument identification is failure in capturing local arguments.

(4) *Tom-ga* *John-no*
Tom-NOM John-*no*
rensho-wo *tome-ta*
consecutive victories-ACC stop-PAST
Tom stopped John's consecutive victories.

In this example, correct argument structure is [REL=*rensho*, NOM=*John*], but the system outputs [REL=*rensho*, NOM=*Tom*] because it matches the entry *X-wo tome-ru* "stop X" and does not consider other arguments in the same noun phrase. We will develop a hierarchical model which looks for local arguments before expanding its search for distant arguments.

5 Conclusion

In this paper we explained the argument structure analysis of event-nouns in Japanese. We described our annotation scheme for event-nouns and defined two tasks for argument structure analysis.

First, we proposed an unsupervised method to collect noun phrase patterns for identifying the event-hood. From our experiments, the precision and recall of determination of event-hood were 73.3% and 80.2%.

Second, we built a machine learning based analysis model of the argument identification task of event-nouns using verb-noun co-occurrences from large corpora. We showed that the co-occurrence information is effective for the argument identification task. Using a case alignment dictionary improves accuracy of nominative and accusative cases.

It is still an open question how we can incorporate a co-occurrence model into the argument structure analyser of event-nouns. In this paper, we combined co-occurrence models and syntactic patterns as features of machine learning methods. However, they could be used in other ways. For example, we could use co-occurrence scores to filter noun candidates for argument identification, and also we could filter arguments with highly accurate syntactic patterns. We plan to further investigate the model of argument structure analysis and extends the model to identify inter-sentential and exophoric arguments of event-nouns.

References

Charles J. Fillmore and Collin F. Baker. 2000. FrameNet: Frame semantics meets the corpus. In *Proceedings of the 74th Annual Meeting of the Linguistic Society of America*.

Yoav Freund and Robert E. Schapire. 1996. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Atsushi Fujita, Kentaro Inui, and Yuji Matsumoto. 2004. Deletion of Incorrect Case Assignments in Automatically Generated Paraphrases of Japanese Light-verb Construction. In *Proceedings of the first International Joint Conference on Natural Language Processing*, pages 14–21.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

Jane Grimshaw. 1990. *Argument Structure*. The MIT Press.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Donald Hindle. 1990. Noun Classification from Predicate Argument Atructures. In *Proceedings of the 28th Annual Meeting of the ACL*, pages 268–275.

Thomas Hoffman. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2005. Anaphora resolution by antecedent iden-

tification followed by anaphoricity determination. *ACM Transactions on Asian Language Information Processing (TALIP)*, 4(4):417–434.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting Syntactic Patterns as Clues in Zero-Anaphora Resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 625–632.

Ryu Iida, Mamoru Komachi, and Kentaro Inui. 2007. Annotating a Japanese Text Corpus with Predicate-Argument and Coreference Relations. In *Proceedings of The Linguistic Annotation Workshop*, pages 132–139.

Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, Akio Yokoo, Hiromi Nakaiwa, Kentaro Ogura, Yoshifumi Ooyama, and Yoshihiko Hayashi. 1997. *A Japanese Lexicon*. Iwanami Shoten.

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic Role Labeling of NomBank: A Maximum Entropy Approach. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case Frame Compilation from the Web using High-Performance Computing. In *Proceedings of the fifth International Conference on Language Resource and Evaluation*, pages 1344–1347.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. 2000. Class-Based Contsruction of a Verb Lexicon. In *Proceedings of the AAAI 00*, pages 691–696.

Taku Kudo and Yuji Matsumoto. 2004. Boosting Algorithm for Classification of Semi-Structured Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 301–308.

Sadao Kurohashi and Makoto Nagao. 1997. Kyoto University Text Corpus Project. In *Proceedings of the third Annual Meeting of Natural Language Processing*, pages 115–118.

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.

Adam Meyers, Ruth Reeves, and Catherine Macleod. 2004a. NP-External Arguments: A Study of Argument Sharing in English. In *Proceedings of the ACL 2004 Workshop on Multiword Expressions: Integrating Processing*, pages 96–103.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004b. The NomBank Project: An Interim Report. In *Proceedings of the NAACL/HLT 2004 Workshop Frontiers in Corpus Annotation*, pages 24–31.

Shinjiro Muraki. 1990. *Nihongo Doushi no Shosou (Aspects of Japanese Verbs)*. Hitsuji Shobo.

National Insitute of Communication Technology. 2002. *EDR Electronic Dictionary Version 2.0 Technical Guide*.

National Language Research Institute. 1964. *Bunruigoihyou*. Shuei publisher. (in Japanese).

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Vladimir N. Vapnik. 1998. *The Statistical Learning Theory*. Springer.

Nianwen Xue. 2006. Semantic Role Labeling of Nominalized Predicates in Chinese. In *Proceedings of the HLT-NAACL*, pages 431–438.

Interpreting Noun Compounds using Bootstrapping and Sense Collocation

Su Nam Kim and Timothy Baldwin
Department of Computer Science and Software Engineering
and
NICTA Victoria Lab
University of Melbourne, VIC 3010 Australia
{snkim,tim}@csse.unimelb.edu.au

Abstract

This paper describes a bootstrapping method for automatically tagging noun compounds with their corresponding semantic relations. Our work takes advantage of the collocation of senses of the noun compound constituents and also word similarity. We exploit this to generate a set of noun compounds from a set of previously tagged noun compounds by replacing one constituent of each noun compound with similar words that are derived from synonyms, hypernyms and sister words.

We started with 200 "seed" noun compounds and generated sets of derived noun compounds with accuracy ranging between 64.72% and 70.78%. We also evaluated the utility of the automatically derived noun compounds when used in combination with existing noun compound interpretation methods.

1 Introduction

Research on the semantics of noun compounds (NCs) has generally focused on either devising systems for interpreting NCs or alternatively automatically interpreting NCs according to a pre-defined set of semantic relations (SRs) (Levi, 1979; Lauer, 1995; Barker and Szpakowicz, 1998). In almost all previous work, the interpretation of semantic relations has been suggested as being highly relevant to NLP applications including question-answering and machine translation.

Linguistically-motivated systems for interpreting NCs vary considerably in the number of semantic relations used (Levi, 1979; Finin, 1980). Computational research on the topic has tended to take a predictably data-driven approach to the question and has combined the development of semantic relation sets with interpretation methods (Vanderwende, 1994; Barker and Szpakowicz, 1998).

There are two primary motivations underlying this work: (1) the suggestion by Jones (1983) and others that disambiguating the word senses of the component words in NCs is an essential component of interpretation, providing the motivation for this research; and (2) the recent trend in NC research towards automated methods that maximize use of small amounts of annotated data, recognizing that annotation is a significant bottleneck in NC research (Barker and Szpakowicz, 1998; Moldovan et al., 2004).

In this paper, we aim to interpret the semantic relations in NCs with a minimum annotation overhead. In this, we differ somewhat from previous research attempts, which have generally presupposed large amounts of training data. As a result, previous methods are not readily portable to new domains or well suited to redeployment over new languages with analogous constructions. Additionally, despite the reliance on large amounts of training data, results have tended to be modest, generally with an error rate of over 40% for representative sets of semantic relations.[1]

The approach discussed in this paper takes NCs such as *apple pie* and *family car*, and attempts to derive an interpretation in the form of the semantic relation between the modifier(s) and the head noun. Examples of such semantic relations include MAKE[2] (e.g. *apple pie* signifies a *pie* made from *apple(s)*) and POSSESSOR (e.g. *family car* signifies a *car* possessed by a *family*).

In our method, we use semi-supervised learning to automatically build up a collection of NCs

[1] Moldovan et al. (2004) claimed about 43% accuracy and Kim and Baldwin (2006) reported an accuracy slightly above 50%.

[2] The semantic relations are taken from Barker and Szpakowicz (1998).

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 129–136,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

tagged with semantic relations, using sense collocations of component words in NCs in the training data. That is, instead of manually tagging all training instances, we tag only a selected handful of seed instances and bootstrap off these by replacing the component words in training NCs with their similar nouns. Similarity is evaluated via ontological semantics, using three basic relation types: synonyms, hypernyms and sister words. Novel NCs which have been generated from training instances via similarity are then tagged according to the semantic relation of the original training instance. In this way, we automatically derive the semantic relations of novel NCs. More importantly, we employ the expanded set of training instances in NC interpretation and find that performance is significantly enhanced.

2 Motivation

Despite the infamous high productivity of noun compounds (c.f. the oft-cited *apple juice seat*), we claim that natural occurrences of NCs are often constrained by the semantic collocation of their constituents. That is, combinations of words from certain semantic classes are more likely to form NCs than others due to their ease of interpretation. Based on this observation, Moldovan et al. (2004) used semantic collocations of constituents to interpret NCs and showed that for NCs with the same sense collocation, the same semantic relation holds with high reliability. Similarly, in Kim and Baldwin (2005), the semantic similarities between training and test instances play an important role in interpreting NCs.

Based on these facts, we hypothesize that not only do given sense collocations in NCs lead to a consistent semantic relation, but semantically similar word combinations share the same semantic relations. For example, the semantic relation of *horse doctor* is OBJECT, as the modifier, *horse*, is acted on by the head noun, *doctor*.[3] When the modifier, *horse*, is replaced by its hypernym, *animal*, we are still able to correctly predict the semantic relation of *animal doctor* as OBJECT. This process can also be seen to apply to sister words. For example, the NC *orange juice* has the semantic relation MAKE that the head noun, *juice* is made from the modifier, *orange*. By replacing *orange* with its sister word, *lemon*, the newly generated NC *lemon juice* also has the same semantic relation, MAKE.

The following examples illustrate how different semantic relations, namely synonym, hypernym and sister words can share the same semantic relation.

(1) $\boxed{\text{automobile}}$ factory

- **Synonym:** car factory
- **Hypernym:** vehicle factory
- **Sister:** truck factory

(2) automobile $\boxed{\text{factory}}$

- **Synonym:** automobile mill
- **Hypernym:** automobile plant
- **Sister:** automobile mint

In (1) and (2), we replace one noun at a time since we want to restrict the degree of semantic variation. In (1) we replace the modifier with related words, and in (2) we replace the head noun with related words. By replacing a constituent with a synonym, we generate *car factory* and *automobile mill*, both of which have exactly the same sense collocation as the original NC (as the replaced constituents belong to the same semantic class as the original words). On the other hand, *vehicle factory*, *automobile plant*, *truck factory*, and *automobile mint*, whose constituents are replaced by a hypernym in the first two instances and sister word in the second two instances, have different sense collocations to that original NC and yet maintain the same semantic relation.

Notice in (2) that with *automobile mint*, we generate what is a pragmatically-marked NC with a semantically-plausible interpretation. That is, our world knowledge about how mints operate and what they produce makes the generated NC sound unnatural, but if we can get beyond this to interpret the NC, the semantic relation we come up with is plausible. To deal with cases like this, our proposed method checks all generated NCs against a pre-compiled list of attested NCs and filters out anything which has not been observed in corpus data.

A second more serious concern is that we will produce an NC which violates our basic assumption about the semantic relation being preserved

[3]The example is taken from Barker and Szpakowicz (1998). Note that defining semantic relations is controversial. In this paper, we do not intend to define or discuss semantic relations. Instead, we decide to use the set of semantic relations in Barker and Szpakowicz (1998) without modification due to reliability shown in previous work.

across controlled semantic variation of the constituent words. For example, an alternate sister of *factory* is *recycling plant*, leading to *car recycling plant*. Here, the most natural interpretation is OBJECT rather than MAKE. We detail a filtering method for dealing with such examples in the next section.

3 Approach

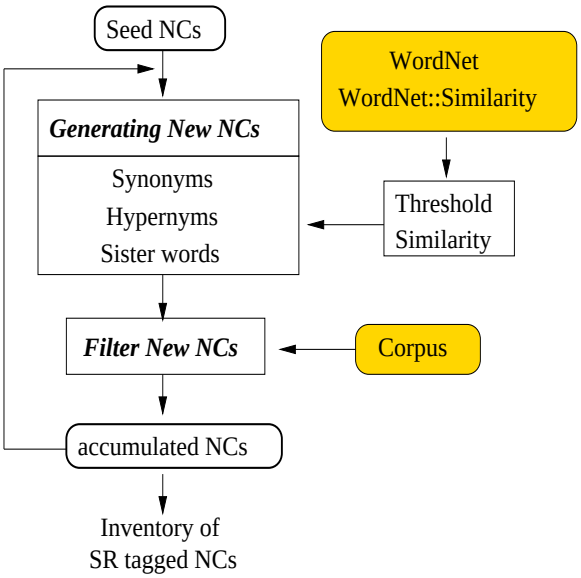


Figure 1: System Architecture

Figure 1 depicts our method for automatically deriving interpreted NCs. We initially begin with a small number of seed NCs that are manually tagged with semantic relations and word senses. We then generate new NCs by replacing each of the modifier and head noun with sense-specified words based on synonyms, hypernyms and sister words derived from WORDNET2.1. Note here that we only replace one noun at a time to avoid over-generation.

Next, we apply a filtering method based on word-level similarity. That is, if the similarity between the original noun and its substitution candidate is less than a predefined threshold, we discard the candidate. Note that the threshold is defined differently for the three ontological relation types. Figure 2 shows the distribution of similarity values across the three ontological relation types, as determined using the WUP (Wu and Palmer, 1994) similarity measure (see Section 4.2) across the cross product of 200 randomly extracted nouns. Table 1 shows the average similarity values and the threshold used in our method for each ontological relation type. The threshold is determined when the number of tested NCs is approximately 90%.

After gathering the generated NCs, we apply a corpus attestation-based filtering method to filter out false positives. Despite the high productivity of NCs, many of the automatically generated NCs are pragmatically or semantically implausible (e.g. *part part* or *activity being*). As we assume there is a default interpretation for a given NC which applies equally to all occurrences of that NC, this check is carried out independent of context.

Finally, we assign the semantic relation from the original NC to the newly-generated NCs derived from it. We then iteratively repeat this procedure over the newly-generated NCs until some termination condition is met, namely no new NCs are generated or a threshold on the number of iterations is met. The number of iterations for synonyms and sister words is limited to 1 since these two relations are commutative.[4] For hypernyms, the only limit on the number of iterations the algorithm could be run over is the depth of the WORDNET hierarchy, but in practice, we stop after the 4th iteration.

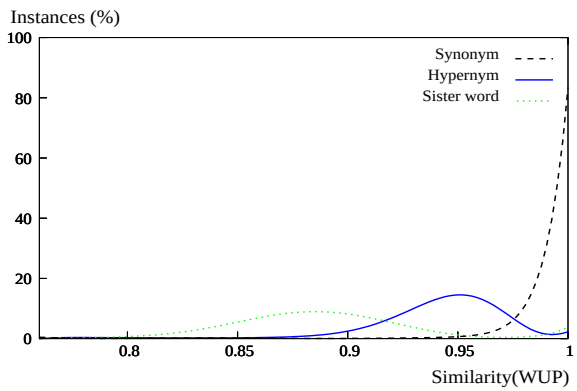


Figure 2: Range of similarity values over different ontological relation types

4 Data and Resources

4.1 Semantic Relations

Semantic relations specify the interpretation of the overall NC relative to it parts. For example, *winter school* is a *school* held during *winter* (semantic relation = TIME). The semantic relation is sensi-

[4]Strictly speaking, they are commutative at the word sense but not necessarily word level.

Relation	Average	Threshold
Synonym	.94	.90
Hypernym	.89	.85
Sister	.83	.80

Table 1: Average similarity and threshold for each ontological relation type

tive to the semantics of the NC parts. Although NCs have similar parts, they could have different semantic relations. For example, in *student discount*, *students* BENEFIT from the *discount*. On the other hand, in *student protest*, *students* are the AGENT of the action represented by *protest*.

As mentioned above, there is no consensus on the optimal number or set of semantic relations for English (or any other language). In this paper, rather than defining a new set of semantic relations, we simply (and naively) use the set of 20 semantic relations defined by Barker and Szpakowicz (1998) as each semantic relation is clearly defined and it has relatively high currency in computational linguistics research. Additionally, since Kim and Baldwin (2005) used the same set of semantic relations, we can directly compare our performance with theirs.

Note that we do not consider the claim that NC interpretation is influenced by the context (Downing, 1977; Warren, 1978) since our work is carried out out of context. Hence, we make the assumption that there is a unique semantic relation for a given sense collocation.

4.2 `WordNet::Similarity`

`WordNet::Similarity`[5] (Patwardhan et al., 2003) is an open source software package developed at the University of Minnesota which facilitates the measurement of semantic similarity or relatedness between a pair of concepts, or alternatively a pair of words. The system provides six measures of similarity and three measures of relatedness based on the WORDNET 2.1 (Fellbaum, 1998) lexical database. The measures of similarity are based on analysis of the WordNet `isa` hierarchy. For our purposes, we used the WUP method (Wu and Palmer, 1994) based on the findings of Kim and Baldwin (2005) for an NC disambiguation task over the same set of semantic relations.

4.3 Data Collection

We randomly gathered and annotated two sets of NCs: a 200 NC seed set and a 400 NC test set. The seed set is for use in automatically generating NCs. The test set (which is disjoint with the seed set) is for use in examining the utility of the NCs semi-automatically derived by our method. That is, we use the second to test the performance of existing interpretation methods over the NC data we generate. The NCs in both sets were sourced from the Wall Street Journal (WSJ) component of the Penn Treebank. Also, there is no overlap between the two data sets. SENSEVAL-2/3 and SemCor were obvious alternate candidates given the sense annotations, but were unsuitable due to the small number of NC types.

Two human annotators tagged both the semantic relations and word senses for the NCs in the two sets and met to resolve any annotation disagreements (these NCs were included in the experiments). Over the two data sets, the initial inter-annotator agreement for tagging semantic relations was 52.3% and the inter-annotater agreement for tagging word senses was 58.8%.

The corpus filter used to exclude bogus NCs uses NCs extracted from the British National Corpus (BNC) and Reuters Corpus using a full text chunker. From these NCs that were extracted, we took only noun-noun bi-grams adjoined by non-nouns (e.g. ... *the apple pie was* ...) to ensure that they were not part of a larger compound nominal. We additionally measured the entropy of the left and right contexts for each noun-noun type, and filtered out all compounds where either entropy value was < 1.[6] This was done in an attempt to, once again, exclude NCs which were embedded in larger multiword expressions, such as *service department* in *social service department*. The combined total number of unique NC types extracted from the two corpora was 389,400.

5 Evaluation

We conducted two different experiments. The first was to automatically derive NCs using the pro-

[5] `www.d.umn.edu/~tpederse/similarity.html`

[6] For the left token entropy, if the most-probable left context was *the*, *a* or a sentence boundary, the threshold was switched off. Similarly for the right token entropy, if the most-probable right context was a punctuation mark or sentence boundary, the threshold was switched off.

		NewNC (A)	PosNC (B)	SRError in (B)	Filtered (C)	Sample (D)	NegNC in (D)	SRError in (D)	AllError in (D)	Accuracy of (A)
Syn	1	384	69	21.74%	315	63	7.94%	27.59%	33.33%	68.75%
	2	229	13	7.69%	216	43	13.95%	18.92%	30.23%	**69.44%**
Hyp	1	616	100	25.00%	516	103	0.97%	30.39%	30.10%	**70.78%**
	2	997	87	22.99%	910	182	6.04%	40.94%	44.51%	57.37%
	3	1,202	70	40.00%	1,132	226	11.06%	44.28%	50.44%	50.25%
	4	1,079	58	43.10%	1,021	204	18.14%	47.90%	57.35%	43.47%
Sis	1	10,728	686	24.05%	10,042	502	7.97%	30.52%	36.06%	**64.72%**
	2	192,917	1,751	30.26%	191,202	956	14.33%	24.79%	35.56%	64.48%
	3	19,275	106	19.81%	19,169	958	7.31%	36.37%	41.02%	59.11%
	4	44	0	0.00%	44	44	68.18%	100.00%	100.00%	0.00%

ROW LABELS:

Syn = synonyms over 1–2 iterations
Hyp = hypernyms over 1–4 iterations
Sis = sister words over 1–4 iterations

COLUMN LABELS:

NewNC (A) = number of newly derived NCs
PosNC (B) = derived NCs positively attested in the corpus
SRError in (B) = ratio of incorrect NCs in (B)
Filtered (C) = NCs not attested in corpus and filtered out
Sample (D) = test sample (sub-sample of (C))
NegNC in (D) = ratio of non-NCs in (D)
SRError in (D) = ratio of incorrectly tagged NCs in (D)
AllError in (D) = total error ratio in (D)
Accuracy in (A) = ratio of NCs tagged correctly in (A).

Table 2: Results for the three ontological relations

posed method. The second was to interpret NCs using existing interpretation techniques including the derived NCs as training data. For the second experiment, we used the interpretation methods introduced in Moldovan et al. (2004) and Kim and Baldwin (2005).

5.1 Experiment I

In this experiment, we tested our proposed method over the three ontological relation types of synonyms, hypernyms and sister words. There are two points of interest in this experiment. The first is how many NCs we are able to derive and the second is how many of the derived NCs are tagged with the correct semantic relation. We test each of these separately for the three ontological relation types.

The results of experiment one are reported in Table 2, for the three ontological relation types over subsequent iterations for each. Here, *NewNC (A)* is the number of newly generated NCs, *PosNC (B)* is the number of NCs remaining after corpus filtering; *Filtered (C)* is the number of NCs filtered out by the corpus filter (note A = B + C); and *Sample (D)* is a sub-sample of instances from (C) for evaluation purposes; and *Accuracy of (A)* is the proportion of correctly generated NCs out of the original (A). Note that negative NCs (NegNC) are pragmatically or semantically infelicitous NCs.

The evaluation is performed from two perspectives. One is to check whether the generated NCs are positively attested in the corpus data. The other is to check the soundness of the predicted semantic relations in the generated NCs. Two human annotators manually checked the correctness of the semantic relations in the newly generated NCs. To evaluate the plausibility of the generated NCs, we categorized the generated NCs into 2 sets based on their corpus attestation: (B) = NCs attested in the corpora, and (C) = NCs not attested in the corpora. The annotators manually checked all the NC instances found in the corpus and a random subsample of the NCs not found in the corpus (due to the resource constraints).

The human annotators checked an average of 180 NCs per hour. The proportion of NCs in (D) tagged with an incorrect semantic relation was computed after removing the NCs not attested in the corpus.

Table 3 shows the cumulative error over subsequent iterations for each ontological relation type.

5.2 Experiment II

The second experiment was aimed at evaluating the utility of the NCs generated by our proposed method. Ideally, we would have liked to have performed extrinsic evaluation over the generated NCs, e.g. in an Machine Translation context. However, due to time limits, we instead only test the generated NCs as additional training data in an NC interpretation task.

In this experiment, we used the NCs generated

		SRError in (B)	SR+NegNCError in (C)	TotalError in (A)
Synonym	1	21.74%	33.33%	31.25%
	2	19.51%	32.08%	30.56%
Hypernym	1	25.00%	30.10%	29.22%
	2	24.06%	39.65%	37.51%
	3	28.40%	44.42%	57.26%
	4	31.11%	48.11%	42.74%
Sister	1	24.05%	36.06%	35.28%
	2	28.49%	35.73%	35.50%
	3	28.12%	37.83%	35.97%
	4	28.12%	38.94%	35.67%

SRError in (B) = cumulative error ratio in (B)
SR+NegNC Error in (C) = cumulative error ratio in (C)
TotalError in (A) = ratio of NCs tagged correctly in (A)

Table 3: Cumulative error in Experiment I

Semantics	Correct	All	ErrorInData2
Synonym	338	813	22.88%
Hypernym	668	3,015	39.90%
Sister	1,042	10,928	34.64%
Combined	1,648	14,356	32.47%

Table 4: Data set sizes in Experiment II

Semantics	Method	seedNC	Correct	All
Baseline	0R	23.00%	–	–
	M+	33.25%	–	–
	K+B	29.75%	–	–
Synonym	M+	–	33.25%	32.50%
	K+B	–	29.00%	29.50%
Hypernym	M+	–	32.50%	32.75%
	K+B	–	**30.50**%	29.00%
Sister	M+	–	**35.50**%	34.25%
	K+B	–	29.00%	28.00%
Combined	M+	–	34.75%	34.00%
	K+B	–	29.75%	29.00%

Table 5: Results of NC interpretation (0R = Zero-R majority vote, M+ = Moldovan et al. (2004), and K+B = Kim and Baldwin (2005))

Semantics	Instances (ErrorRate)	M+	K+B
Manual	1769 (0.00%)	–	42.00%
Synonym	613 (30.34%)	32.50%	29.50%
Hypernym	616 (29.32%)	31.75%	**32.75%**
Sister	686 (24.05%)	29.00%	28.00%

Table 6: Results of interpretation with a fixed amount of training data, with and without noise in the data

by our method as training data within existing interpretation methods. In this, we are interested in: (1) how well the methods operate using the generated NCs as training data relative to the size and nature of data; and (2) given similar amounts of training data and error rates, whether data derived from the different ontological relation types leads to different interpretation results. In this, we use two previous methods: Moldovan et al. (2004) and Kim and Baldwin (2005).

For the first test, we used two data sets: (1) all *correct* data (including seed NCs) derived using the three ontological relation types, i.e. all NC instances that were manually verified to be correct (*Correct*); and (2) all generated NCs from (B) (including the seed NCs once again) irrespective of correctness (*All*). In the second instance, we took all NCs generated over all iterations from synonyms and hypernyms. Since the number of NCs derived using sister words were large, we decided to use only the ones generated in the first iteration. The data set sizes are shown in Table 4. We use a zero-R majority class method as the baseline for the interpretation task. [7]

[7] The majority of instances in the test set have the TOPIC semantic relation

6 Discussion

In the first experiment, we predictably derived the largest number of NCs via sister words. After the first iteration, from the 200 seed NCs, we generated 384, 616 and 10,728 NCs using synonyms, hypernyms and sister words, respectively. Even when we used only the positive NCs found in the corpus (i.e. strict attestation filtering), the numbers of newly generated NCs are 69, 100, and 686, respectively. As such, we can see that our method is successful in automatically deriving a large number of NCs.

When we looked at the error rate of the newly derived NCs, we observed that the NCs generated via synonyms contain less errors than the other two ontological relations. However, from an interpretation point of view, the accuracies in using hypernym and sister word are relatively high compared to accuracies reported for previous methods, namely 43.2% for M+ (Moldovan et al., 2004) and 53% for K+B (Kim and Baldwin, 2006), as compared to between 64.7% and 70.8% after the first iteration of our method. In addition, our method needs only a small number of manually tagged NCs (200 NCs in this experiment) compared to

prior approaches. As a result, we strongly believe that our method can efficiently derive NCs tagged with semantic relations with a lower error rate and reduced human effort. Also, our method has the potential to significantly increase the volume of automatically interpreted NCs.

Note that when we use hypernyms and sister words we go beyond the scope of the original semantic collocation, so that we can significantly increase the number of tagged NCs by iterating repeatedly. In Table 2, we can see that the increase in error rate as we get further and further away from the original semantic collocations is gradual (linear). Also, in the filtered (unattested) NCs, the error increases gradually, and that in excluding these in our method, the error rate is kept down, especially with synonyms and sister words. Considering that the number of generated NCs is exponential, especially with hypernyms and sister words, the total error rate is relatively low. Finally, the results in Table 3 show that the error rate for NCs derived via synonyms and sister words is linear even after several iterations. However, we believe that although we used two filtering methods for reducing noise in the derived data sets, we require more efficient methods to effectively avoid noise in generated data. We leave this as an area for future work.

The second experiment in Table 5 was to test the utility of the generated NCs for existing interpretation methods. Although the number of NCs generated by our method is large, the range of sense collocations we capture is restricted, especially for synonyms. The baseline for each method was to use only the 200 seed NCs as training data, which in itself outperformed a simple majority class baseline for both methods tested. Using additional training data generated by our method, we were able to improve over this baseline performance. Although the number of generated NCs is 5 times more than the 200 seed NCs, the performance of Moldovan et al. (2004) does not increase significantly. This confirms that idea that Moldovan et al. (2004) relies not only on the amount of training data but also on the range of sense collocations in the training data.

We also analyzed the performance of the two methods using error-free (Correct) and "error-full" (All) training data. Despite the smaller amount of training data, the best performance is achieved using the Correct data in Moldovan et al. (2004). This is disappointing in terms of the general-purpose NC interpretation task targeted in this paper, but very promising for domain-specific NC interpretation. Our reasoning here is that the filtering could be tailored to a given domain, such that we could generate NCs with higher lexical similarity to NCs in that domain, and hence a higher chance of producing a correct classification.

Finally, we evaluated the performance of interpretation methods in terms of the three different ontological relations over similar amounts of training data. In Table 6, the manual performance uses the 1769 hand-tagged training instances. We were unable to tag word senses for all of the training data. As a result, we have shown the performance for only Kim and Baldwin (2005). Using the three different ontological relations, we constructed similar amounts of training data. All the training data included only corpus-attested NCs with correctly tagged semantic relations. Although we achieved the best performance from hypernym semantics using Kim and Baldwin (2005), the difference in performance is not significant. As a result, we conclude that there is no significant difference between the three relation types with respect to the task of NC interpretation.

7 Conclusion

In this paper, we proposed a novel method for automatically deriving NCs tagged with semantic relations using bootstrapping and sense collocations. We used synonyms, hypernyms and sister words as the ontological relations for the substitution of constituent in NCs. We were able to achieve between 64.72% and 70.78% accuracy in deriving NCs and we found that the error rate while deriving NCs increased nearly linearly when using synonyms and sister words. We also demonstrated the utility of the derived NCs as training data, and concluded that the performance of interpreting NCs depends not only on the amount of training data but also the sense collocations in the training data. The main concern of this approach is to interpret NCs within the same or similar scope of sense collocation. As future work, we will further exam the proposed method to expand not only interpreted NCs but also the variation of sense collocations.

References

Ken Barker and Stan Szpakowicz. 1998. Semi-automatic recognition of noun modifier rela-

tionships. In *In Proceedings of the 17th international conference on Computational linguistics*, pages 96–102.

Pamela Downing. 1977. On the creation and use of English compound nouns. *Language*, 53(4):810–842.

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

Timothy Wilking Finin. 1980. *The semantic interpretation of compound nominals*. Ph.D. thesis, University of Illinois, Urbana, Illinois, USA.

Karen Sparck Jones. 1983. *Compound noun interpretation problems*. Prentice-Hall, Englewood Cliffs, NJ, USA.

Su Nam Kim and Timothy Baldwin. 2005. Automatic interpretation of compound nouns using wordnet::similarity. In *In Proceedings of 2nd International Joint Conference on Natual Language Processing*, pages 945–956. Jeju, Korea.

Su Nam Kim and Timothy Baldwin. 2006. Interpreting semantic relations in noun compounds via verb semantics. In *In Proceedings of COLING/ACL*, pages 491–498. Sydney, Australia.

Mark Lauer. 1995. *Designing Statistical Language Learners: Experiments on Noun Compounds*. Ph.D. thesis, Macquarie University.

Judith Levi. 1979. The syntax and semantics of complex nominals. In *New York:Academic Press*.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. 2004. Models for the semantic classification of noun phrases. In *HLT-NAACL 2004: Workshop on Computational Lexical Semantics*, pages 60–67.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2003. Using measures of semantic relatedness for word sense disambiguation. In *In Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*.

Lucy Vanderwende. 1994. Algorithm for automatic interpretation of noun sequences. In *In Proceedings of the 15th conference on Computational linguistics*, pages 782–788.

Beatrice Warren. 1978. *Semantic Patterns of Noun-Noun Compounds*. Ph.D. thesis, Actr Universitatis Gothoburgensis.

Zhibiao Wu and Martha Palmer. 1994. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133 –138.

Semantic Analysis of Entity Contexts towards Open Named Entity Classification on the Web

Xuan-Hieu Phan, Susumu Horiguchi
Graduate School of Information Sciences, Tohoku University
hieuxuan@ecei.tohoku.ac.jp
susumu@ecei.tohoku.ac.jp

Le-Minh Nguyen
Graduate School of Information Science, Japan Advanced Institute of Science and Technology
nguyenml@jaist.ac.jp

Cam-Tu Nguyen
Faculty of Information Technology, College of Technology, Vietnam National University, Hanoi
ncamtu@vnu.edu.vn

Abstract

This paper introduces the use of Latent Semantic Analysis (LSA) to uncover semantic structures/concepts hidden in entity contexts towards improving named entity recognition (NER) on the Web. The underlying idea of the paper is that words surrounding entities of the same category are potentially related to each other in one way or another. Analyzing such relations helps build implicit concepts around entity types, making entity contexts more discriminative, and avoiding data sparsity for a better classification. Our experiments on a Web data collection of entity contexts show that semantic analysis can give a significant error reduction.

1 Introduction

Most NER systems (Borthwick 1999; Florian et al. 2003; McCallum and Li 2003) commonly take text sentences as input sequences and recognize named entity chunks as outputs. The classifiers in those systems usually focus on local information, i.e., syntactic and lexical features, and sometimes using external lookup dictionaries and gazettes in order to determine entity boundary and type. Also, most systems deal with MUC (Grishman and Sundheim 1996) or CoNLL (Sang and Meulder 2003) datasets with a small set of predefined entity categories, such as person, location, and organization. These datasets are suitable for method evaluation than producing practical NER systems.

Towards building real-world applications, such as entity-oriented search, online question answering, and information extraction on the Web, recent methods (Pasca 2004; Etzioni et al. 2005; Banko et al. 2007; Downey et al. 2007) have been focusing on much more complex tasks regarding heterogeneous, less grammatical, and noisy Web text collections with a broader range of entity types. And, recent studies (Downey et al. 2007) show that the single use of supervised sequential learning models (McCallum et al. 2000; Lafferty et al. 2001) is not enough to achieve high accuracy on this kind of data. To be more successful on the Web, NER systems should seek for more appropriate techniques that can take advantage of the nature of Web data such as bootstrapping, un-/semi-supervised learning (Collins and Singer 1999; Etzioni et al. 2005), co-occurrence statistics (Downey et al. 2007), etc.

Inspired by the latter trend, our work attempts to analyze implicit semantic concepts hidden in entity contexts that would be useful for building practical NER systems on the Web. Our main observation is that words surrounding entities of the same type are potentially relevant to each other semantically in one way or another. For instance, two named entities *J2SE* and *Oracle*, both belong to the *product & technology* category, might occur in two following contexts [..., install, 1.4.2, standard, edition, ...] and [..., installation, version, 9i, ...]. We can see that words from such entity contexts implicitly form semantic concepts like {install, installation, setup, ...} and {edition, version, release, beta, ...}. These concepts potentially describe different properties of entity types. And thus, in our viewpoint, finding these concepts from data is possibly useful for named entity type identification.

One might also argue that some classifiers like maximum entropy can deal well with the data sparsity problem, and building those concepts is not necessary for classification. However, our motivation includes more important targets. First, semantic analysis of entity contexts means that we can do some global statistics over the whole dataset. This is even more important to large collections of Web data. As claimed in (Downey et al. 2007), many long and complex named entities are

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 137–144,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

too ambiguous and difficult for correct recognition if using only local classifiers. However, statistics over a large Web collection, in this case, can help. Semantic analysis, in this sense, can make entity contexts more discriminative for a better recognition. And we can see NE context classification is performed in a collective manner in this way. Second, several powerful kernel-based classification methods fail to achieve high accuracy due to the vector sparsity, that is, unable to get good kernel similarity. In this case, semantic analysis helps reduce the original sparse and high-dimensional lexical space to a much lower-dimensional concept space for a better similarity measurement.

In addition to the semantic analysis of entity contexts, this paper presents a less-expensive framework to apply this idea to the Web data that requires a little supervision and is quite convenient for developing practical, open-domain information extraction systems.

The remainder of the paper is organized as follows. Section 2 presents the semantic analysis of entity contexts with LSA. Section 3 describes entity context classification with SVMs. The experimental evaluation and discussion are presented in Section 4. Section 5 reviews some related work. And, conclusions will be given in Section 6.

2 Semantic Analysis of Entity Contexts

2.1 Entity Contexts and Concept Space

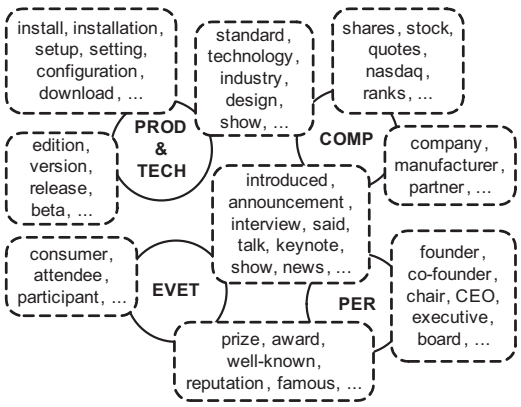


Figure 1: An example of ideal relationship between clusters of words and semantic concepts regarding named entity categories (PROD & TECH - product & technology; COMP - company; PER - person; EVET - event)

Let $\mathbf{D} = \{X_1, X_2, \ldots, X_N\}$ be a data collection of N named entity contexts in which each context $X_i = [w_i^{-k}, \ldots, w_i^{-1}, \underline{e_i}, w_i^1, \ldots, w_i^k]$ includes a maximum of $2k$ word tokens around a named entity e_i, i.e., words are chosen within a window with a size of $2k$. To reduce noise, special characters and stop words are removed from the contexts. Let n be the total number of words/tokens occurring in $\mathbf{D}$, each context X_i can be represented as a sparse bag-of-word vector $\mathbf{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ in the vector space $\mathbf{R}^n$. Table 1 shows some examples of named entity contexts.

Let $\mathbf{C} = \{C_1, C_2, \ldots, C_m\}$ denotes a set of m concepts which the entity contexts in $\mathbf{D}$ are related to. Semantic analysis for entity contexts is to find a map or transformation s from $\mathbf{R}^n$ to the concept space $\mathbf{R}^m$ as follows.

$$s : \mathbf{R}^n \rightarrow \mathbf{R}^m \tag{1}$$

or, in other words, for each context $X \in \mathbf{D}$, the map s helps reduce $\mathbf{x}$ to a vector $\mathbf{y}$ in the concept space.

$$s : \mathbf{x} = \{x_1, x_2, \ldots, x_n\} \rightarrow \mathbf{y} = \{y_1, y_2, \ldots, y_m\} \tag{2}$$

where each y_i ($1 \leq i \leq m$) measures the proximity or relevance between the entity context X and the concept $C_i \in \mathbf{C}$.

Figure 1 depicts an ideal relationship between lexical information and semantic concepts potentially hidden in a data collection. Each dash rectangle is a cluster of words/tokens that indicates a semantic concept related to one or more named entity categories. This relationship/map is what we expect to estimate from data for a better named entity classification. Theoretically, the semantic map s that we desire to build should satisfy two important conditions:

- Words/tokens that are related to each other semantically (e.g., synonyms or relevance in meaning) in the lexical space tend to be highlighted and converged to the same concept in the concept space.
- The resulting concept space is much lower-dimensional compared to the original lexical space.

The first condition is most critical because it is at heart of the semantic transformation defined in (1) and (2). However, semantic analysis methods, in practice, never achieve this goal completely due to several reasons, for example, noise in data and, of course, automatic transformation never attains human-level observations. The second condition

Named entity contexts	Type
checking iPod shuffle battery charge	PROD
restoring Pod shuffle factory settings	PROD
iPod play content purchased itunes	PROD
apple iPod nano product	PROD
MS Office 2007 beta version	PROD
MS Office version 2007	PROD
summary MS Office offers simple protect document	PROD
protect MS Office documents strong encryption aes	PROD
add strong encryption feature MS Office protecting word excell powerpoint	PROD
disallowing choose MS Office limiting usage products based	PROD
listed links weblogs reference MS Office expensive	PROD
Intel introduced quad-core processors including	COMP
Intel today announced design completion	COMP
january 16 2007 Intel today announced fourth-quarter revenue	COMP
Intel largest semiconductor manufacturer world	COMP
shares stock quotes Intel nasdaq gs intc ranks	COMP
500 million debt offering Softbank shares doubled u.s.	COMP
views customer requirements practices Oracle industry standard technologies	COMP
company research find information Oracle operations financials officers	COMP
heard campus Steve Jobs 2005 commencement address	PER
listen address Steve Jobs ceo apple computer pixar	PER
board directors complete confidence Steve Jobs senior management team	PER
apple publicly replace co-founder Jobs chose chief operations officer	PER
they showing highlights past Gates keynotes 10th year he	PER
Gates keynote focused key issues	PER
acm fellows award Donald E. Knuth	PER
Tim Berners-Lee inventor web award knighthood	PER
hoped steve jobs announce Macworld Expo keynote address tomorrow	EVET
autodesys showcases form.z 6.1 Macworld Expo macminute news	EVET
technology demo showcased booth Consumer Electronics Show	EVET

Table 1: Examples of named entity contexts

is quite easy to be satisfied because most semantic mapping techniques also include dimensionality reduction.

There might be different approaches and methods, e.g., fuzzy-based or statistics-based, to analyzing semantic information in count data in general and text/web data in particular. And we, of course, can apply any mathematical tool provided that it would be able to uncover semantic relations and concepts in the data collection to achieve a more accurate entity type classification. We decided to use Latent Semantic Indexing/Analysis (LSI/LSA) for its popularity and simplicity.

2.2 Latent Semantic Analysis

LSA, first introduced in Deerwester et al. (1990), has been applied in text indexing and information retrieval over the past decade. This indexing technique aims to highlight (semantic) proximity between data items via their co-occurrence and, more importantly, second-order co-occurrence like "*a* and *b* do not co-occur; X mentions term *a* in context C and Y mentions *b* in context C; $\rightarrow$ *a* and *b* are somehow related to each other". Therefore, LSA is usually used to address synonyms or terms (conveying a synonymity association) that tend to occur in the same, similar or related contexts.

Mathematically, LSA relies on singular value decomposition (SVD), a well-known factorization method in linear algebra. Since its introduction more than 40 years ago SVD has become a standard decomposition technique. It is a great technique for uncovering and emphasizing hidden

or "latent" data structures while removing noise. Technically, SVD helps transform and reduce the original sparse lexical space to a dense and much lower-dimensional concept space.

In practice, the resulting lower-dimensional space is not exactly the same as the theoretical concept space we expect to build although it somewhat satisfies the properties of an ideal semantic transformation.

2.3 Semantic Analysis of Contexts with LSA

In order to transform the original lexical vector space into a concept space, we first form the word-context matrix $\mathbf{A}$ of size n x N (i.e., there are N contexts and each is represented in the n-dimensional vector space) as follows.

$$\mathbf{A} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1N} \\ x_{21} & x_{22} & \dots & x_{2N} \\ . & . & \dots & . \\ x_{n1} & x_{n2} & \dots & x_{nN} \end{pmatrix}$$

where x_{ij} ($i = 1..n$ and $j = 1..N$) is the occurrence frequency of word w_i in context X_j. We do not use other measures like TF-IDF because each entity context contains only a few words. This matrix is very similar to term-document matrix in information retrieval except that it is much sparser because entity contexts are much shorter than text documents.

In order to increase the first and second-order co-occurrence of words in the matrix, each named entity e_i itself is included in its context vector $\mathbf{x}_j = \{x_{1j}, x_{2j}, \dots, x_{nj}\}$. For example, named entity like iPod is counted in its vector in order to emphasize the correlation among iPod's contexts in the whole data collection. This is a bit restricted because it somewhat violates the nature of data. However, including named entities in the word-context matrix makes contexts of the same entity type more correlated as well as helps reduce the data sparsity in the matrix.

To build the semantic transformation, SVD first decomposes the matrix $\mathbf{A}$ into three matrices $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$ as follows.

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{3}$$

where the columns of $\mathbf{U}$, called *left singular vectors*, are the eigenvectors of $\mathbf{A}\mathbf{A}^T$; the columns of $\mathbf{V}$, called *right singular vectors*, are the eigenvectors of $\mathbf{A}^T\mathbf{A}$; and $\mathbf{S}$ is a diagonal matrix. By definition the nondiagonal elements of diagonal matrices are zero. The diagonal elements of $\mathbf{S}$ are a special kind of values of the original matrix. These are termed the singular values of $\mathbf{A}$.

Each singular value in $\mathbf{S}$ is interpreted as one dimension in the new space. And singular values are usually sorted in descending order. Higher-order values (i.e., larger values) are more important than lower-order ones. Choosing only first k singular values in $\mathbf{S}$ means we reduce the new space to k-dimensional. This can be seen as analogous to the PCA dimensionality reduction, that is, entity contexts will be mapped to a lower-dimensional latent semantic space induced by selecting directions of maximum covariance (Baldi et al. 2003). And, it is equivalent to the following approximation.

$$\mathbf{A}_k = \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^T \tag{4}$$

The new reduced space is interpreted as semantic concept space in the language of semantic indexing in IR. To transform a vector $\mathbf{x}$ from the original lexical space $\mathbf{R}^n$ into its corresponding vector $\mathbf{y}$ in the k-dimensional concept space, we follow the following equation.

$$\mathbf{y} = \mathbf{x}^T\mathbf{U}_k\mathbf{S}_k^{-1} \tag{5}$$

where $\mathbf{S}_k^{-1}$, the inverse matrix of $\mathbf{S}_k$, is computed easily because $\mathbf{S}_k$ is a diagonal matrix.

3 Entity Context Classification

Let $\mathbf{D}_r = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ and $\mathbf{D}_s = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$ be the training and testing datasets consisting of N and M vectors in lexical space $\mathbf{R}^n$, respectively. The semantic transformation is built on the training data as described in the previous section to obtain $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$. Selecting k dimensions, we get $\mathbf{U}_k$, $\mathbf{S}_k$, and $\mathbf{V}_k$ to form a k-dimensional concept space.

Then both training and testing datasets will be mapped into the concept space to obtain $\overline{\mathbf{D}}_r = \{\overline{\mathbf{y}}_1, \overline{\mathbf{y}}_2, \dots, \overline{\mathbf{y}}_N\}$ and $\overline{\mathbf{D}}_s = \{\overline{\mathbf{y}}_1, \overline{\mathbf{y}}_2, \dots, \overline{\mathbf{y}}_M\}$ using the following equation.

$$\overline{\mathbf{y}} = \overline{\mathbf{x}}^T\mathbf{U}_k\mathbf{S}_k^{-1} \tag{6}$$

where $\overline{\mathbf{x}}$ is obtained from $\mathbf{x}$ by discarding the occurrence of the named entity (i.e., by setting the count of named entity to zero). For example, we remove iPod, Intel, or Steve Jobs from their context vectors. This is very important because we only examine and classify entity contexts without entity names.

Technically, we can use any classifier to evaluate the benefits of semantic analysis. However Support Vector Machines is probably a suitable choice because of its powerful ability to deal with real-value vector via kernel functions.

4 Experiments

4.1 Data and Experimental Settings

Our dataset consists of named-entity contexts of four categories, *company* (COMP), *event* (EVET), *person* (PER), and *product & technology* (PROD). In order to retrieve the data, we used our Web crawling toolkit (JWebPro[1]) to perform search requests to Google automatically according to the names listed in Table 2. We used these famous names because we want to guarantee that top-400 or 500 returned documents of each search query only refer to the same real-world entity. As a result, the entity contexts extracted from those documents all belong to the same category. And, we also used different name expressions (e.g., "IBM", "IBM Corp.", and "International Business Machines") for a particular name to retrieve as many as entity contexts as possible. This is the way we built a *labeled* dataset of thousands entity contexts from the Web within only several hours.

Number of training entity contexts	5,714
Number of testing entity contexts	1,904
Vector space dimension	11,489

Table 3: Dataset statistics

The dataset[2] we used for evaluation has 7,618 entity contexts that was randomly divided into two parts: 5,714 for the training set and 1,904 for the test set. The former is used for semantic analysis and the latter is used for testing only. The dimension of vector space built from this dataset is 11,489. This means that the data is very sparse.

We used SVDLIBC[3], a C library for singular value decomposition, to perform semantic analysis on the training set and obtained the vectors of singular values with different sizes/dimensions ranging from k = 10 to 800. Then, both the training and test sets were transformed to the concept spaces according to those singular vectors.

We used LIBSVM[4], a well written multiclass SVM implementation, to train and classify entity contexts in both cases: (1) original vector space as a baseline and (2) concept space (i.e., with semantic analysis). We trained the training set using all four common kernels (linear, polynomial, radial basis function - RBF, and sigmoid). For each kernel, we performed parameter selection using the tool provided by LIBSVM to achieve as high accuracy as possible. All kernels showed the accuracy improvement of semantic analysis but with slightly different extents. The experimental results reported in Figures 2, 3, and Table 4 were obtained using radial basis kernel. The linear kernel gave a little lower improvement while the two remaining kernels provided lower results.

4.2 Results

We performed named entity context classification with two experimental settings: (1) without and (2) with SVM parameter selection in order to verify two assumptions: semantic analysis can help reduce data sparsity for a better kernel measure and, more importantly, semantic analysis can improve classification accuracy for named entity contexts. In both cases, we took the classification without semantic analysis as a baseline for comparison.

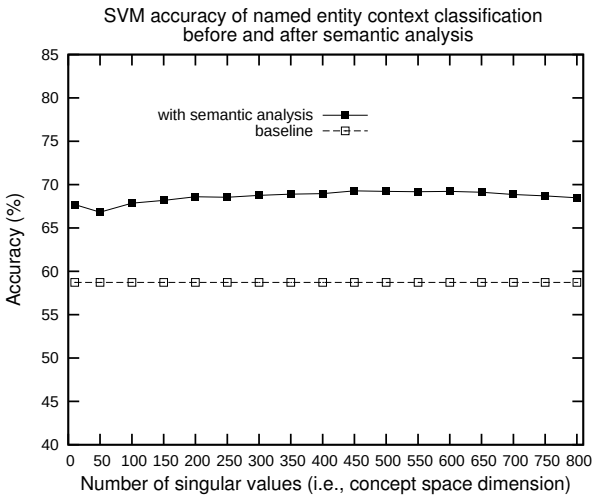


Figure 2: Accuracy comparison without parameter selection for SVMs

Figure 2 shows the classification accuracy of the semantic analysis and the baseline without SVM parameter selection. We chose different concept space dimensions (i.e., the number of singular values) varying from 10 to 800 in order to examine how the number of singular values influences the

[1] JWebPro: http://jwebpro.sourceforge.net/
[2] http://jwebpro.sourceforge.net/PACLING2007Data.tar.gz
[3] SVDLIBC: http://tedlab.mit.edu/∼dr/SVDLIBC/
[4] LIBSVM: http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

NE Types	Searched Names
Company	Google, IBM, Intel, Microsoft, NTT, Novell, Oracle, Samsung, Softbank, Sony, Sun Microsystems, Toshiba, Yahoo
Event	Consumer Electronics Show (CES), FIFA World Cup, Formula One, Macworld, Summer & Winter Olympic Games, World Expo
Person	Larry Page, Sergey Brin, Steve Ballmer, Bill Gates, Steve Jobs, Masayoshi Son, Tim-Berners Lee, Donald Knuth
Product & Tech	Google Earth, Google Video, Java, MS Access, MS Excel, MS FrontPage, MS Office, MS Word, Outlook Express, SQL Server, Oracle, Mac OS, iPhone iPod, iTune

Table 2: Searched names for Web data preparation

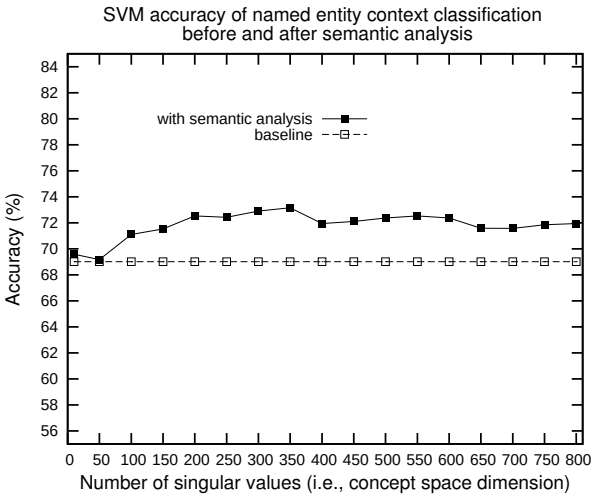


Figure 3: Accuracy comparison with parameter selection for SVMs

SVM parameter tuning:	No	Yes
Baseline accuracy	58.72%	69.01%
Sem. analysis accuracy	69.28%	73.16%
Error reduction	**25.6%**	**13.4%**

Table 4: Best accuracy and error reduction comparison (both two cases: without and with parameter tuning for SVMs)

classification accuracy. Also, this is the way to tune an appropriate concept space dimension for a particular dataset. Classification in sparse lexical space gives a baseline accuracy of 58.72%. This is quite lower due to the sparsity and the imbalance in the dataset. We looked in the test results and saw that almost all output labels fall into the major class (PROD - *product & technology*). Classification after semantic analysis provide accuracy of around 69%, significantly higher than the baseline's. This means that our semantic analysis can transform the original sparse lexical space into a much lower–dimension and dense space with a significant improvement in accuracy. Table 4 reports the best accuracy improvement (from 58.72% to 69.28%) and error reduction (25.6%) achieved when the number of singular values is 450.

Figure 3 shows the results in the case that we attempted to seek for appropriate SVM parameter values (i.e., values for C and γ if using RBF kernel). We used the parameter selection tool provided by LIBSVM to evaluate on the training set and find out preferable values. In this case, SVM can deal with the sparsity and imbalance in the data and get a much higher baseline accuracy, 69.01%. However, we also achieved higher classification accuracy with semantic analysis for almost all concept space dimensions. The third column in Table 4 reports the highest accuracy achieved when the concept space dimension equals 350. The biggest error rate reduction in this experimental setting is approximately 13.4%. This means that semantic analysis can really provide useful evidence for a significant improvement in entity context classification.

4.3 Discussion

NER systems usually use all possible information to achieve high recognition accuracy. For example, the named entity "Oracle Corp." itself is discriminative enough to be recognized as a company (COMP) because of the appearance of an informative term like "Corp.". Similarly, "iPod" will easily be classified as a product (PROD) if we maintain an external look-up dictionary for common and famous product names. However, in the case that a named entity neither contains informative

terms nor appears in look-up dictionaries, the context around it become a unique source of evidence for classification. There are a huge number of such named entities on the Web. They are not common enough to appear in training data or to be stored in pre-built look-up dictionaries. Another type of error is that some common named entities are highly ambiguous. For example, the entity "Lincoln" in the context [... get free new Lincoln prices, invoice pricing ...] is a car model even though it might be stored as a personal name in look-up dictionaries. Again context information (e.g., "free", "prices", "invoice") around it is the key evidence for entity type classification.

NER for such above-mentioned named entities become hard. Our semantic analysis framework aims at resolving these difficult names. Building a concept space for an entity context data collection makes the entity contexts more discriminative for classification. This is the way semantic analysis helps improve named entity recognition on the Web.

By looking into the collection of contexts, we observed that stemming is probably important to semantic analysis. We saw a lot of tokens/words that refer to the same concept or closely related in meaning like {show, showing, shows, showed}, {introduce, introduced, introduction}, or {award, awarded, awards}. And the data spareness is in part derived from this phenomenon. We believe that performing word stemming before analysis would result in better performance, and this will be studied in the future work.

Another point we would like to emphasize is the way we build labeled collection of entity contexts from the Web for semantic analysis. This only needs a little initial supervision. All the things we did is to choose a list of common named entities and then search them from the Web. Also, we can extend the range of named entity types easily. And this is quite convenient for practical open named entity recognition and/or information extraction.

5 Related Work

There have been different approaches to NER since its introduction. (Statistical) machine learning techniques deal with NER by taking text sentences as input sequences and recognize named entity chunks (Borthwick 1999; Florian et al. 2003; McCallum and Li 2003). They use various kinds of information including contexts, word/token itself, external look-up dictionaries and gazettes provided that the systems are as high accurate as possible. There were also several shared tasks and standard datasets like MUC (6, 7), CoNLL (2002, 2003), and ACE for common evaluation and competition for choosing best systems. However, there is little attention to linguistic phenomena related to this task like multiword expressions, word/term co-occurrence, and collocation statistics. Also, most NER studies focused on close-domain and standard datasets for the main purpose of method evaluation rather than moving to a more open-domain and heterogeneous text environment like the Web.

Our work, on the other hand, focuses on analyzing semantic structures/concepts hidden in named entity contexts in order to examine how this kind of information influences the entity type classification. We target to the Web environment because it is much more complex than any plain text collection in terms of the diversity of text genres and noisy information. More importantly, there is no universal dictionary or gazette that can cover all possible named entities on the Web. They are too diverse and belong to various domains. As a result, contexts around named entities become one of the most significant sources of evidence for entity type recognition. This is probably the biggest motivation for us to study semantic analysis of contexts.

Probabilistic and statistical modeling of text and Web data is one of important trends in current NLP, IR, and data mining research. In additional to LSA/LSI, several statistical and semantic models like Latent Dirichlet Allocation (Blei et al. 2003), Correlated Topic Models (Blei and Lafferty 2005) were introduced to model the (semantic) relation among words, topics, and documents; and applied in citation matching, entity resolution (Bhattacharya and Getoor 2006). Our work, semantic analysis of entity contexts, can be seen as an instance of this trend, that is, finding hidden concept relations among entity contexts in a Web data collection.

6 Conclusions

We have presented a semantic analysis framework that attempts to find out useful, implicit semantic correlations between words/tokens of named entity contexts for a more accurate context classification. The semantic transformation from lexical vector space to the concept space makes entity

contexts more discriminative and, especially suitable for classification using kernel-based methods. Experimental results assure that semantic information hidden in named entity contexts is worth considering for named entity recognition in particular and information extraction on the Web in general.

In the future, we will evaluate our method on a larger set of entity contexts with a broader range of domains and entity types to examine the benefits of semantic analysis towards building practical information extraction systems.

Acknowledgements

This work is fully supported by the research grant No.P06366 from Japan Society for the Promotion of Science. We would like to thank anonymous reviewers for their useful comments and nice suggestions for the future work.

References

P. Baldi, P. Frasconi, and P. Smyth. 2003. *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*. Wiley.

M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. *Open Information Extraction from the Web*. In Proc. of IJCAI-2007.

M.W. Berry. 1992. *Large-Scale Sparse Singular Value Computations*. Journal of Supercomputer Applications, 6(1):13-49.

I. Bhattacharya and L. Getoor. 2006. *A Latent Dirichlet Model for Unsupervised Entity Resolution*. In Proc. of SIAM SDM-2006.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. *Latent Dirichlet Allocation*. Journal of Machine Learning Research, 3:993-1022.

D.M. Blei and J. Lafferty. 2005. *Correlated Topic Models*. In Proc. of NIPS-2005.

A. Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. PhD Thesis, New York University.

M. Collins and Y. Singer. 1999. *Unsupervised Models for Named Entity Classification*. In Proc. of EMNLP-1999 and VLC-1999.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. 1990. *Indexing by Latent Semantic Analysis*. Journal of the American Society for Information Science, 41(6):391-407.

D. Downey, M. Broadhead, and O. Etzioni. 2007. *Locating Complex Named Entities in Web Text*. In Proc. of IJCAI-2007.

O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. *Unsupervised Named Entity Extraction from the Web: An Experimental Study*. Artificial Intelligence, 165(1):91-134.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. *Named Entity Recognition through Classifier Combination*. In Proc. of CoNLL-2003.

R. Grishman and B. Sundheim. 1996. *Message Understanding Conference 6: A Brief History*. In Proc. of COLING-1996.

T. Hofmann. 1999. *Probabilistic Latent Semantic Analysis*. In Proc. of UAI-1999.

J. Lafferty, A. McCallum, and F. Pereira. 2001. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. In Proc. of ICML-2001.

T.A. Letsche and M.W. Berry. 1997. *Large-Scale Information Retrieval with Latent Semantic Indexing*. Information Sciences, 100(1-4):105-137.

A. McCallum, D. Freitag, and F. Pereira. 2000. *Maximum Entropy Markov Models for Information Extraction and Segmentation*. In Proc. of ICML-2000.

A. McCallum and W. Li. 2003. *Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons*. In Proc. of CoNLL-2003.

M. Pasca. 2004. *Acquisition of Categorized Named Entities for Web Search*. In Proc. of CIKM-2004.

M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. 2006. *Names and Similarities on the Web: Fact Extraction in the Fast Lane*. In Proc. of COLING-ACL-2006.

E. Riloff and R. Jones. 1999. *Learning Dictionaries for Information Extraction by Multi-level Bootstrapping*. In Proc. of AAAI-1999.

T.K. Sang and D. Meulder. 2003. *Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition*. In Proc. of CoNLL-2003.

Recognizing Names by the Meaning of Defining Syntactic Structures

Li Yang and Steven Abney
Department of Linguistics
University of Michigan
{lyshane|abney}@umich.edu

Abstract

We present a named entity recognition system that relies on interpreting the meaning of the syntactic structures that define named entities. The syntactic heads of the defining structures are learned from the training data and are later used to interpret the meaning of the syntactic structures in the test data. Semantic features are extracted based on the interpretations and are used to build a decision tree classifier. The mapping phase iteratively matches previously labeled names with aliases and conjunctions in the same document. Evaluation on two test sets shows this system can generalize to some extent and perform moderately well.

1 Introduction

The sixth Message Understanding Conference (MUC-6) separated the named entity recognition (NER) task from other tasks of Information Extraction and formally defined this task for the Information Extraction community. From MUC-6 to present, a number of systems have been dedicated to the NER task. The authors of the systems often found themselves trying to answer the essential questions raised by those participating in MUC-6. Three of these questions are: first, how can information inside and in the external context of names be used? Second, how can non-local structures that are other mentions of the same name in a document be handled? And third, what syntactic and semantic information can be incorporated and how it can be used?

The rule-based MUC-6 and MUC-7 systems provide satisfying answers to the first question by pattern matching using hand-built rules. Rule-based systems handle the second problem with string matching methods.

The majority of the systems developed since MUC-6 are statistical learning systems. The statistical systems effectively utilize the internal and external evidence of entities. For example, for the CoNLL-2003 shared task Chieu and Ng (2003) introduced the idea of global information that captures useful corpus-level contextual information surrounding names, such as the bigrams to the left of names. Some statistical systems model the non-local structures in a document. For example, Finkel et al. (2005) achieved non-local consistency by integrating label consistency constraints with a CRF model using Gibbs sampling.

Thus, the first two questions do not seem to pose a challenge for either the rule-based systems or the statistical systems. However, statistical systems have not provided or do not intend to provide an answer for the third question. For example, the only syntactic features Finkel et al. (2005) included are part-of-speech tags. While drawing on semantic patterns expressed in the manually-built rules, rule-based systems do not employ higher level syntactic and semantic information (MUC-6, 1995; MUC-7, 1998). As a matter of fact, the NYU system for MUC-6 chose not to use full sentence parses (Grishman, 1995). Since semantic analysis follows syntactic analysis in the NYU system, lacking syntactic parse indicates absence of semantic analysis. The NYU system for MUC-6 promised to come back to syntax and semantics in future work.

Despite the fact that most statistical systems use minimum syntactic and semantic information, a small number of systems experimented with combining syntactic and semantic information with statistical methods. Cucchiarelli et al. (1998) are among the pioneers that utilize syntactic relations such as subject-object, and noun-preposition-noun, and head word information in the NER system. The first step of the system by Niu et al. (2003) extracts five types of verb-

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 145–153,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

based syntactic structures and builds a decision list classifier with the structures. Stevenson and Greenwood (2005) build an unsupervised system using the automatically extracted subject-verb-object patterns.

In this paper, we propose an answer to the third question by presenting a system that mainly relies on several types of syntactic structures that define the modifying or modified names. The system also relies on the syntactic structures highly correlated with names. The system proceeds by iteratively mapping the class labels assigned to names in earlier steps to names in coordinate structures and to multiple mentions of the same name in the same document. We emphasize the role of interpreting the meaning of names found in the modifying or modified syntactic configurations in our work and choose the decision tree as our base classifier. Our goal is to test how the syntactic and semantic information we utilize contributes to the NER task. We also test how our semantic approach generalizes when given different test data sets.

2 Structures Defining Names

We begin with the structures that provide for definitions for named entities.

2.1 Syntactic Heads Defining Names

After a careful study of multiple corpora for Information Extraction, such as those from MUC-6, MUC-7, ACE, CoNLL-2003 etc., we observed the following syntactic structures whose heads define the corresponding names.

Left Appositive In the following example, the head noun **firm** of the left appositive *the market-research **firm*** provides the defining description of the organization *InfoCorp*.

> *... said Sandy Gant, an analyst at the market-research **firm** InfoCorp.*

Right Appositive In the previous example, the head noun **analyst** of the right appositive *an analyst at the market-research **firm** InfoCorp* gives the defining description of the person name *Sandy Gant*.

In the following example, the head word **insurer** in the right appositive *the largest home and auto **insurer** in California* gives the defining description to the organization *State Farm Mutual Automobile Insurance Co.*

> *State Farm Mutual Automobile Insurance Co., the largest home and auto **insurer** in California...*

Internal Heads The head noun *avenue* in the NP *Fifth Avenue* indicates that this is a location name. Similarly, the head noun *service* in the NP *U.S. Postal Service* indicates that this NP is an organization name. There are two special cases with regard to name internal heads. In the first case, a name consists of only a single word. We treat the single-word name as the head of itself. In the second case, some names do not possess a syntactic head, for example the person name *Bertrand Russell*. In this case, we treat the full name as the head.

Right Modified Noun Head The modified head noun *subsidiary* of *Southwide Inc.* indicates that *Southwide Inc.* is an organization.

> *...said Murray Robinson, president of Delta & Pine Land Co., a Southwide Inc. **subsidiary**...*

Morphology defining names To complete our list of structures that define names, we include the suffixes that define the type of a name. For example, the suffix *-Bank* of *CitiBank* gives the defining description of the organization *CitiBank*.

2.2 Co-occurring Structures

In addition to the above syntactic structures that define names explicitly, one can deduce the name types of the NPs in the following two types of syntactic structures based on the meaning of the structures.

Left Modified Noun Head The modified head noun *president* in the NP ***president** of Delta* indicates that the NP *Delta* inside the modifying PP *of Delta* is an organization.

Verb-P-NP Structure The verb-P-NP structure *flew to Dallas* in *He flew to Dallas on Oct. 19...* indicates that ***Dallas*** is a location.

2.3 Coordination

Since coordinate structures connect phrases of the same syntactic type, names in coordination structures belong to the same name class.

2.4 Aliases

If a name belongs to class *c*, other mentions of the same name and its variations in the same document belong to the same class *c*.

2.5 Related Work

We follow the approaches of previous systems in using some of the structures discussed above. MUC-6 and MUC-7 systems assign the same class label to other mentions of the same name in a document (MUC-6, 1995; MUC-7, 1998). Names in a coordinate structure also belong to the same class in MUC-6 and MUC-7 systems. Collins et al. extracted the right appositives and left modified heads along with the names for their unsupervised models (Collins and Singer, 1999).

3 Requirements for Recognizing Names with Definitions

With the structures discussed in section 2 in mind, we are ready to discuss how definitions contribute to name recognition on the sentence level and the document level. The steps leading to name recognition by utilizing the meaning of the head words in the structures discussed above on the sentence level are summarized as follows:

1. Identify a noun phrase as a candidate name.
2. Identify a noun phrase that modifies or is modified by the candidate phrase.
3. Identify the head noun of this phrase.
4. Identify the head noun of the candidate phrase.
5. Determine the class of the candidate name based on the meaning of the head nouns identified in previous two steps.

These steps apply to defining configurations described in section 2.1. As for the co-occurring structures, we can apply the same steps to the case of left modified noun heads without modification. For the verb-P-NP structure, in addition to applying the preceding steps, we also check the meaning of *verb*.

On the document level where names repeat, if one of the repeats is correctly interpreted and its name class correctly determined from its definition(s) if present, then we can assign the same class to other occurrences of this name.

From the above, we summarize several requirements for utilizing the meaning of the modifying and modified configurations of names, which include parsing, constituent analyses from the parsing results, knowledge about the semantics of the noun heads of definite descriptions, knowledge about the meaning of the head of the co-occurring structure, and identifying the name candidates (the segmentation task). Recognizing names in coordinate structures also shares these requirements. We show how these requirements are satisfied in section 5.

4 The data

Training data We use the training data for the CoNLL-2003 shared task as our training data. The original training data for the shared task were given in one file (CoNLL-2003, 2003). We split them into 945 files according to the document boundaries in the files because our system treats each document as a training/test unit. 158 of the 945 files are tables listing score results from various sports events. These files have been filtered out using our vector-space document classifier. The reason for filtering out the tables is that many data entries in tables are not sentences or phrases. In later stages of our system, all data that have not been parsed before will be parsed with the Charniak parser [1]. Syntactic and semantic analyses will be carried out on the parsed data. It would not make much sense to parse these table entries and perform syntactic and semantic analyses on them. The knowledge base, discussed in section 5.3.1, is constructed from the same set of training data.

Test data We labeled the organization, person, and location names for sections 23 and 24 of the Penn Treebank. Section 24 is used as our development data. Section 23 is used as the test data. We also use the MUC-6 dry-run data set as our second test data set. We did not use the CoNLL-2003 shared task development and test data for development or testing for two reasons. First, our goal is to test how well our semantic approach can generalize across different test data. Second, there are tables in the data, which our current system does not handle.

5 Meeting the Requirements

Section 3 lists the requirements to recognize names with defining information. Components of our system that meet the requirements are introduced in this section.

[1] The 2006 reranking version available at ftp://ftp.cs.brown.edu/pub/nlparser/

5.1 Preprocessing

Given a training or test document, the preprocessor proceeds as follows:

1. Parse the document with the Charniak parser.
2. Run a modified version of the chunklink script [2] over the parsed document so that the constituents and their head information are extracted.
3. If the input is a development or test document, run the name segmentation module so that all name candidates are extracted. If the input is a training document, read names from the document.
4. Run a syntactic analyzer so that all syntactic structures associated with a name candidates is extracted. The structures are those listed in section 2.

5.2 Name Segmentation Module

Candidate names need to be identified for the development and test data. This requirement is met by the segmentation module, which relies on both heuristic and probabilistic methods. On the heuristic side, we rely on the following information:

- NP phrase boundaries detected by the parser
- case information
- POS information from the parser
- knowledge about the left appositives (section 5.3.1)
- top-ranked unigrams preceding names learned from the training data
- knowledge about time expressions

The probabilistic method resolves the issue of judging whether a preposition is part of the name spanning from the preceding noun when it appears between two initially capitalized nouns. Specifically, if the probability of the bigram formed by the preposition and the preceding noun being part of a name of any class is greater than 0, then the preposition is part of the name spanning from the preceding noun and should not be separated from it. Stated formally, this is, if $p(w_{i-1}, w_i, t) = \frac{\#(w_{i-1},w_i,c)}{\#(w_{i-1},w_i)} > 0$, where w_i is a preposition, for any $c \in nameClasses$, then w_i is part of the name spanning from w_{i-1}. An example on how the segmentation module uses this method to decide whether the noun phrase *Organization for Internet Safety* is a candidate name is given below.

During development or testing, suppose the Charniak parser produces a correct parse for the phrase *Organization for Internet Safety* as follows (the noun phrases are indexed for convenient referencing):

$(_{NP_0}$ $(_{NP_1}$ *Organization)* $(_{PP}$ $(_{P}$ *for* $(_{NP_3}$ *Internet Safety*)))).

When this parse is handed down to the segmentation module, it cannot yet decide whether the noun phrase NP_0 is a candidate name due to the embedded PP in NP_0. To determine if NP_0 is a candidate, the segmentation module checks if the probability of the triplet $(Organization, for, c)$ is greater than 0, where c is any name class. To calculate $p(Organization, for, c)$, we need to know two types of counts, namely $\#(Organization, for, c)$ and $\#(Organization, for)$. So, if $p(Organization, for, c) = \frac{\#(w_{i-1},w_i,c)}{\#(w_{i-1},w_i)} = \frac{\#(Organization,for,c)}{\#(Organization,for)} > 0$, then the segmentation module returns $NP0$ as a candidate name. The counts $\#(w_{i-1}, w_i, c)$ and $\#(w_{i-1}, w_i)$ are stored in a data structure.

5.3 Representing Meaning of Head Words

Sections 2 and 3 illustrate that determining the class label of a name depends on the meaning of the head(s) in the modified or modifying syntactic structure(s). We represent the knowledge about the meaning of the heads by collecting them into categories corresponding to the syntactic structures described in section 2. All categories are then stored in a linguistic knowledge base.

5.3.1 Building the Knowledge Base

The linguistic knowledge base is learned automatically from the training data.

If a syntactic structure is a definite description of its modifying or modified name, collect the head noun to a semantic category corresponding to this syntactic structure, along with the class label of the name. For example, the head noun **firm** of the left appositive *the market-research **firm*** provides definite description for the organization *InfoCorp*. Then the head noun **firm** is stored in the semantic category called **left appositives**, along

[2] Original code by Sabine Buchholz available at http://ilk.uvt.nl/ sabine/chunklink/README.html

with the **organization** class label. The semantic categories listed in row two through five of table 1 are learned in a similar manner.

The probability of each left modified noun head and the preposition of the modifying PP co-occurring with a class label is calculated, using the formula $p(left_modfd_N_head, prep, c) = \frac{\#(left_modfd_N_head, prep, c)}{\#(left_modfd_N_head, prep)}$. The top n noun heads for each class are stored in the linguistic knowledge base, along with the preposition and class label. Row six of table 1 is such an example.

The probability of the verb-P in a verb-P-NP structure co-occurring with the class label of the NP when the NP is a name is calculated, using the formula $p(verb-P, c) = \frac{\#(verb_P, c)}{\#(verb_P)}$. The top n verb-P's for each class are stored in the linguistic knowledge base, along with the class label. The *fly-to* verb-P structure in row seven of table 1 is an example.

Category Name	Head Word	Class	Freq.
left appositives	firm	ORG	n_1
right appositives	analyst	PER	n_2
internal heads	Service	ORG	n_3
internal heads	Avenue	LOC	n_4
right modified noun	subsidiary	ORG	n_5
left modified noun	president-of	ORG	n_6
Verb-P-Adverbial	fly-to	LOC	n_7
morph-ending	-Bank	ORG	n_8

Table 1: Learned Sample Entries in Knowledge Base

The frequencies of each item entered the knowledge base are also recorded. A sample linguistic knowledge base is built based on the examples from section 2. Table 1 shows this sample knowledge base.

5.3.2 Other Information in Knowledge Base

Following Chieu and Ng (2003), our linguistic knowledge base also stores other types of global information, such as bigrams of all names from the training data, a list of common time expressions including days and months, and a list of US states and their old and new abbreviations. The last two lists are the only external resources we provide for the system.

5.3.3 Related work

Our linguistic knowledge base collects the semantic information from the syntactic structures modifying or modified by names. And that is our approach to representing the contextual information about names. The method we calculate the probability of the headword co-occurring with a name type is similar to Chieu and Ng (2003).

Talukdar et al. (2006) discuss a method to learn contextual information from a fixed n window size preceding and following a name. Signal words are selected statistically from the extracted context using a term-weight method. Since we perform syntactic analyses in order to find the syntactic structures that define names, we do not limit ourselves to any given fixed window size. This could be a more flexible approach when a defining head word appears at a distance from the name that is beyond the fixed window size of n.

6 System Design

Having introduced the components that prepare the system, we sketch our system architecture below. The main procedures of the system proceed as follows:

1. Preprocessing (5.1)
2. Segmenting candidate names (5.2)
3. Building linguistic knowledge base (5.3)
4. Extracting features (6.1)
5. Classification A supervised classifier built from the semantic features from the **feature extractor** is used to classify the test data. The specific classifier chosen is the J4.8 decision tree classifier from the Weka package (Witten and Frank, 2005).
6. Mapping Names (6.2)

The feature extraction module and the mapping phase that handles aliases/non-local dependencies and conjunctions are highlighted in the following.

6.1 Feature Extraction

The feature extractor performs semantic analyses and extracts semantic features. The feature extractor is fed with the linguistic knowledge base and a document. It then iterates through every sentence in the document. For each candidate in a sentence, it scans through all the syntactic structures modifying or modified by the candidate name, while it sets the following features for the candidate:

clsLeftAps or noLeftAps If the candidate has a left appositive and the head noun of the left appositive is found in the linguistic knowledge base with the class label **cls**, then

clsLeftAps is set for the candidate name. Otherwise, **noLeftAps** is set. For example, since the head noun **firm** of the left appositive *the market-research **firm*** in *an analyst at the market-research **firm** InfoCorp.* is found in the knowledge base with the class label **ORG** in table 1, the **orgLeftAps** feature is set for the name candidate *InfoCorp*.

clsRightAps or noRightAps If the candidate has a right appositive and the head noun of the right appositive is found in the linguistic knowledge base with the class label **cls**, then **clsRightAps** is set for the candidate name. Otherwise, **noRightAps** is set. For example, since the head noun **analyst** of the right appositive *an analyst* in ... *said Sandy Gant, an analyst...* is found in table 1 with the class label **PER**, the **perRightAps** feature is set for the name candidate *Sandy Gant*.

clsIntHd or noIntHd If the head of a candidate name is found in the linguistic knowledge base with the class label **cls**, then **clsIntHd** is set. Otherwise, **noIntHd** is set. For example, the **locIntHd** feature is set for *Fifth Avenue* since **Avenue** is found in table 1 with the class label **LOC**.

clsRMHd or noRMHd If the head of the right modified noun phrase of a candidate name is found in the linguistic knowledge base with the class label **cls**, then **clsRMHd** is set for the candidate. Otherwise, **noRMHd** is set. For example, the **orgRMHd** feature is set for *Southwide Inc.* since **subsidiary** is found in table 1 with the class label *ORG*.

clsLMHd or noLMHd If the left modified noun head concatenated with the preposition of the modifying PP that contains the candidate name is found in the linguistic knowledge base with the class label **cls**, then **clsLMHd** is set for the candidate name. If the candidate name is in the verb-PP-adverbial structure and the verb concatenated with the preposition is found in the linguistic knowledge base with the class label **cls**, then **clsLMHd** is set for the candidate name. Otherwise, **noLMHd** is set for the candidate name. For example, the **orgLMHd** feature is set for *Delta* since **president-of** it is found in table 1 with the class label *ORG*. And the **locLMHd** feature is set for *Dallas* since **flew-to** is found in table 1 with the class label *LOC*.

clsMorphEnd or noMorphEnd If the candidate name has an ending that shows up in the linguistic knowledge base with the class lable **cls**, then **clsMorphEnd** is set for the candidate name. Otherwise, **noNorphEnd** is set. For example, since the *-Bank* ending of *CitiBank* appears in table 1 with the class lable **ORG**, the **orgMorPhEnd** feature is set.

6.2 Mapping Names

The decision tree classifier trains on the semantic features created by the feature extractor. It then labels the test data that are also represented as the semantic features described in section 6.1. Given the names labeled by the decision tree classifier, the name-mapping module proceeds iteratively in each document as follows:

1. Collect the previously labeled names, either by the classifier or through the following steps.

2. If one of the aliases of the same candidate names in the document is labeled as c, then the same class label c is assigned to the rest of the aliases. Aliases or other mentions of a candidate name are identified by full or partial string/substring matching.

3. Find candidate names in coordinate structures. If one of the names in conjunction is labeled as class c by a previous step, then label the rest of the candidate names in the coordinate structure as c.

4. The mapping module then repeats until the class distribution resulting from the previous steps among the labeled data does not change. The class distribution from the classifier and the mapping module is defined as: $p(c) = \sum_{i=1}^{k} p(c_i)$, where k is the number of classes.

6.2.1 Related Work

Several systems have taken a multi-step approach to handling other mentions of the same name in a document. Being one of the top performing NER systems on MUC-7, the LTG System (Mikheev et al., 1998) proceeds in five phases. Like other rule-based systems, the first phase and

third phase match the input document with hand-built rules and dictionaries. In the second and fourth phases, the names that are labeled from the preceding phase and their other occurrences and variations are collected as potential names. A Maxent model then determines the probability of these names being actual names. Niu et al. (2003) reported a two-stage system where the second stage builds an HMM model based on the labeled names from the first stage. A more recent two-stage system whose second stage is explicitly dedicated to handling aliases was reported by Vijay and Manning (2006). This system builds upon the CRF system of Finkel et al. (2005) by adding a second stage that utilizes features derived from the names labeled from the first stage.

7 Experiments and Results

The system has been trained on the training data for the CoNLL-2003 shared task. Section 24 of the Penn Treebank was used as the development data. Section 23 of the Penn Treebank and the MUC-6 dry run data were used as the test data sets. The training data and the MUC-6 dry data were parsed by the Charniak parser. The reason we use the Penn Treebank data as the test data is to minimize the system's dependence on the automatic parser. The training data and the two test data sets are chosen from different resources. By doing so, we try to maximize the differences between the training data and test data. At the same time, our goal is to test if we can generalize from the syntactic configurations that define names.

7.1 Segmentation Results

Segmentation is an important phase in our system, which identifies the candidate NP phrases for the classification and normalization phases. Table 2 shows that the recall scores for the two test data sets are 96.53% and 92.76% respectively. The recall scores are higher than the precision scores. This is an expected result because the segmentation module identifies all the NPs in a document, some of which belong to name classes other than person, location, and organization names that we are interested in for our current experiments. For example, the segmentation module finds *Pay-Cut Plan* as a candidate name in the first document of the MUC-6 dry run data, but it is not an organization, location, or person name. Table 2 also presents the actual number of correct segmentations and the total number of names in both data sets.

Data	Prec.	Rec.	ttlCrct	ttlEnts
Sec. 23	77.02	96.53	2785	2885
MUC-6	74.66	92.76	769	829

Table 2: Segmentation Results

7.2 Labeling Results

Table 3 lists the results from the base classifier and those from the mapping process. For the Treebank data, the F-measure goes from 75.9% by the base classifier to 85.01% after the normalization. For the MUC-6 data, the F-measure rises from 66.3% to 82.40%. The improvement in F-measures shows that the system design is helpful during both the classification stage and the mapping stage. Given that the training and test data sets we used are not from a training/test set designed for some specific NER task like the MUC NER task or the CoNLL-2003 shared task, the increase in F-measures indicates that generalizing across different corpora based on the meanings of names can be done to some degree. However, the results in terms of F-measure are moderate. We would like to analyze the errors on the MUC-6 dry run data in section 8.

	Decision Tree			Name Mapping		
Data	Prec	Recl	F1	Prec	Rec	F1
Sec. 23	90.5	65.3	75.9	87.35	82.78	85.01
Muc-6	84.5	54.5	66.3	84.43	80.46	82.40

Table 3: Classification & Mapping Results

7.3 Comparisons with Other System(s)

Most of the NER systems we have reviewed since 2003 are trained and tested on the training and test data for the CoNLL-2003 shared task. In section 4 we explained that we chose not to test our system on the CoNLL-2003 test data set because syntactic and semantic analyses cannot be carried out on tables. Other systems that we have reviewed since 2003 experimented with data sets that we were not able to obtain. As a result, we are unable to report on any direct comparisons with the NER systems in literature since 2003. The closest comparison we could make is the one with the system of Chieu and Ng (2003). Being one of the top NER systems for the CoNLL-2003 shared task, Chieu and Ng's system is known for its novel use of global information and sophisticated feature engineering.

This system was used to annotate the Penn Treebank data for the CoNLL-2005 shared task of semantic role labeling.

Following the CoNLL-2003 labeling format, Chieu and Ng's system annotates all the sections in Treebank with five classes of labels, namely *person, location, organization, and misicillaneous*. The annotations are distributed along with the CoNLL-2005 training/test data sets. We found Chieu and Ng's system comparable to ours for the following reasons. First, we perform one of our two final tests on section 23 of the Treebank. Section 23 is among the sections annotated by Chieu Ng's system. Second, Chieu and Ng's system is originally trained on all the CoNLL-2003 training data, and our system is trained on a subset of the CoNLL-2003 training data, excluding the documents that are tables. Table 4 compares the performance of the two systems over three name classes, *LOC*, *ORG* and *PER* on section 23 of the Treebank.

	Chieu & Ng			Yang & Abney		
Class	Prec	Recl	F1	Prec	Rec	F1
LOC	76.9	88.1	82.1	89.6	81.1	85.2
ORG	82.0	76.1	78.9	85.9	84.9	85.4
PER	58.9	71.0	64.5	88.9	78.9	83.6
All	75.3	78.2	**76.7**	87.6	82.8	**85.01**

Table 4: Results by Name Class

Table 4 shows our current system has some edge over Chieu and Ng's system. We would like to bring to the reader's awareness the following differences between the two systems. First, the two systems rely on different architectures. Chieu and Ng employ the maximum entropy model as the base learning model and do not follow the training-testing paradigm with any post-processing steps. We use the decision tree as the base learning model and the training and testing procedures are followed by a series of iterative name mapping steps. Second and more importantly, we experiment with syntactic features that provide for good representation of the names.

8 Error Analysis and Future Work

As we summarized above, the system generally does a moderately good job in terms of taking advantage of structures that define names and mapping labeled names to aliases and conjuctions by matching. However, as shown in table 5 of the performance breakdowns by name class, there is a lot of room for improvement.

	Decision Tree			Name Mapping		
Class	Prec	Recl	F1	Prec	Rec	F1
LOC	73.7	69.7	71.6	73.5	68.97	71.1
ORG	85.2	43.0	57.2	86.2	82.96	84.5
PER	94.5	80.7	87.1	88.6	82.67	85.5
All	84.5	54.5	66.3	84.43	80.46	82.4

Table 5: Results by Name Class

In order to understand how our system can be improved, we performed error analyses on the results from the MUC-6 dry run data. We made this decision because we do not plan to use this data set for testing purposes in any future experiments. In the following we summarize the three main types of errors we found in the system output.

Names Without Defining Structures This is the case where names do not have definitions about them and the linguistic knowledge base has not learned any information about these names from the training data. For example, the system missed labeling person name *Donald Trump* in *The Journal's reference to Donald Trump's reputed desire...* because the name does not have any defining information about it.

Finding an effective approach to handling the names without definitions is next on our research agenda. One way we could go about this is to train a high precision statistical system that learns directly from the context without syntactic parsing and semantic analyses and then combine it with our definition-based system.

Segmentation Errors Another important source of misclassifications is segmentation errors. One type of these is names containing a preposition. Even though we developed a probabilistic boundary identifier (section 6.2) to determine whether a preposition should be part of the name spanning from the previous word, the identifier sometimes still misses the boundaries because it has not seen the bigram consisting of the previous noun and the current preposition in the training data. For example, the boundary identifier splits the organization name *Communications Workers in America* into two candidate names *Communications Workers* and *America* because it has not seen an example of the bigram *Workers in* being part of a name in the training data and hence assigns a zero probability to *Workers in* for being part of a name.

We realize that the segmentation module relies to a large extent on how much training data it has seen before and on how similar the test data for-

mat is to the training data. One way to go about this is to feed the segmentation as much training data as possible. And, at the same time it might be beneficial to make the segmentation module as flexible as possible.

Restricting the Matching During Name Mapping The overall precision score from running the MUC-6 data only moved down about 0.07% from the decision tree classification to the mapping process. This does not lend us a good reason for worrying about the operations during name mapping. However, the overall precision from running Section 23 of the Treebank data dropped about 3.15% shows that we have to restrict the matching during name mapping. For example, in one of the development documents, *Nintendo Co.* was correctly labeled as an organization name. During the mapping stage, the system incorrectly labeled *Nintendo* as organization due to its partial match with *Nintendo Co*. However, this decision was made without checking the context of *Nintendo* that is *Nintendo buyers have little money...*, where *Nintendo* refers to the *Nintendo Entertainment System*.

As we explained in section 6.2, the mapping module assigns the same class label of a labeled candidate name to candidate names that fully match or partially match with the labeled candidate name. The mapping process does not check the context of the partially or fully matched candidate name, which may suggest a different name class as in the preceding example. As future work, we will consider adding a statistical component that checks the probability of an alias being assigned the same class label as the candidate name. This is similar to the approach adopted by the LTG system (Mikheev et al., 1998).

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0308024. The authors would like to thank the anonymous reviewers for their valuable comments. The authors would also like to thank Kevin McGowan and Terrence Szymanski at the Michigan CL (Computational Linguistics) group for their useful comments on the paper.

References

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

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110. SIGDAT.

CoNLL-2003. 2003. *Proceedings of CoNLL-2003*. ACL, Edmonton Canada.

Alessandro Cucchiarelli, Danilo Luzi, and Paola Velardi. 1998. Automatic semantic tagging of unknown proper names. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 286–292. ACL.

Jenny Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. ACL.

Ralph Grishman. 1995. The nyu system for muc-6 or where's the syntax? In *Proceedings of the 6th conference on Message understanding*, pages 167–176.

Andrei Mikheev, Claire Grover, and Marc Moens. 1998. Description of the ltg system used for muc-7. In *Proceedings of the 7th Message Understanding Conference*, pages 1–12.

MUC-6. 1995. *Proceedings of the 6th conference on Message understanding*. Morgan Kaufmann, San Mateo.

MUC-7. 1998. *Proceedings of the 7th conference on Message understanding*. MUC-7, Fairfax, Virginia.

C. Niu, W. Lei, J. Ding, and R.K. Srihari. 2003. A bootstrapping approach to named entity classification using successive learners. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 335–342. ACL.

M. Stevenson and M. Greenwood. 2005. Semantic approach to ie pattern induction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 379–386. ACL.

P. Talukdar, T. Brants, M. Liberman, and F. Pereira. 2006. A context pattern induction method for named entity extraction. In *Proceedings of the 10th Conference on Computational Natural Language Learning*, pages 141–148. CoNLL-X.

Krishnan Vijay and Christopher D. Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1121–1128. ACL.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, San Diego.

A FrameNet Approach to Connectives: The Polysemy of *While*

Satoru Uchida
University of Tokyo
Komaba 3-8-1, Meguro-ku, Tokyo, Japan
JSPS Research Fellow
uchida@phiz.c.u-tokyo.ac.jp

Seiko Fujii
University of Tokyo
Komaba 3-8-1, Meguro-ku, Tokyo, Japan
sfujii@boz.c.u-tokyo.ac.jp

Abstract

This study examines the polysemous meanings of the connective *while*, proposing an extended FrameNet approach for the descriptions of connectives. The meanings of connectives are described with respect to the two frames that each of the conjoined clauses evokes, whose combinational patterns are termed "frame valences." The analysis of the core 50 complex sentences featuring *while* revealed 10 major frame valences. We will demonstrate that laying out such frame valences is a useful method for differentiating between the four senses of *while* on the one hand, and for clarifying the differences between *while* and its synonymous connective *although* on the other. The list of frame valences obtained in this study will be further tested with the data (250 tokens) randomly drawn from the British National Corpus (BNC).

1 Introduction

This study aims to clarify the meanings of connectives[1] by scrutinizing the combinational patterns of semantic frames evoked in each of the two clauses conjoined with the connective. This study is based on FrameNet[2] (Baker et al., 2003; Fillmore et al., 2003), in that the meanings of clauses are described in light of the semantic frames evoked by the participating lexical items, and the core data of this study are either derived from the annotated example sentences of FrameNet or use FrameNet dictionaries to supplement clauses that had yet to be annotated with the specifications of evoked frames. This study, however, is an extension and application of the FrameNet framework to connectives that have not been fully described in the present FrameNet.

Although many studies have been conducted on connectives (e.g., Schiffrin, 1987; Sweetser, 1990; Blakemore, 1987, 2000), few attempts have been made to explicate the types of semantic entities that are actually conjoined by connectives. In one such attempt, Mann and Thompson (1986, 1988) proposed the Rhetorical Structure Theory (RST), which defines the types of relations realized by connectives in the form of conceptual descriptions of the nucleus (which is typically realized in the main clause) and the satellite (which is typically expressed in the subordinate clause). For example, in the RST, "concession" is defined as a relation that links a nucleus describing a "situation affirmed by the author" and a satellite describing a "situation which is apparently inconsistent but also affirmed by the author." This description precisely characterizes the conceptual relations between the conjoined clauses in abstract terms; however, it does not explicitly specify exactly which semantic entities are represented by each of the clauses. The description of the satellite (i.e., the situation which is apparently inconsistent) is relative to the role of the nucleus, which is indeed an important aspect of the semantic connection expressed by a concessive connective. However, the origin of the notion of being "inconsistent" and exactly what types of clauses can be conjoined have been left unexplained. Based on the descriptions of the given clauses that are objectively defined, this would make it difficult to determine the types of relations that the conjoined clauses bear.

The present study proposes an extended FrameNet approach to descriptions of connectives that can provide an algorism that determines

[1] Connectives refer to linguistic items such as *and*, *because*, *but*, and *so*. Researchers have called them by diverse cover terms: discourse markers by Schiffrin (1987), pragmatic markers by Fraser (1996), discourse particles by Schourup (1985), and so on. In order to avoid confusion, we will use the term connectives as a theory-neutral general term in this paper.

[2] http://framenet.icsi.berkeley.edu/.

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 154–162,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

the relations realized by a connective, based on the specifications of frames evoked by the given clauses. We postulate that it is the frames evoked in the main and subordinate clauses that are combined by a connective. Thus, we describe the meanings of connectives with respect to the two frames evoked by each of the conjoined clauses. In our study, such combinational patterns of frames observed with respect to a particular connective are termed "frame valence." To demonstrate and examine this approach, the present study focuses on one English connective, namely, *while*. It also attempts to describe the polysemy of *while* in terms of the frame valence. Furthermore, based on this approach, we will briefly discuss the possibility of differentiating between the meanings of synonymous connectives by using the concessive *while* and *although* as examples.

This paper is organized as follows. Section 2 will provide an overview of FrameNet and discuss how connectives are treated in the current FrameNet descriptions that leave connectives unanalyzed for the most part. Section 3 will explain our proposal for applying the FrameNet framework to the descriptions of connectives. Section 4 will examine *while* as a case study of this extended framework, presenting 10 types of frame valences of *while* based on the core data. Section 5, in turn, will evaluate the list of frame valences of *while* obtained in Section 4, using a larger set of data randomly drawn from the BNC.[3]

2 FrameNet and its descriptions of connectives

This Section provides an overview of FrameNet. First, we consider how words are defined in the dictionary and introduce some key concepts. Then, we discuss how connectives are treated in the current version of FrameNet.

2.1 An overview of FrameNet

FrameNet is an online dictionary based on Frame Semantics. The dictionary differs from other ordinary dictionaries in that words are defined in relation to *semantic frames*. Semantic frames are human knowledge or beliefs about the world that are related to frequently occurring situations and evoked by the use of lexical items in language (Fillmore, 1982; Bednarek, 2005). In FrameNet, such frames are assumed to be evoked by lexical items such as verbs, nouns, and adjectives. For example, the verb *buy* evokes the Commerce_buy frame defined as "describing a basic commercial transaction involving a buyer and a seller exchanging money and goods, taking the perspective of the buyer." As the definition indicates, semantic frames contain elements such as *buyer* and *goods*, and these are called frame elements. There are three types of frame elements, namely, core, non-core, and extra-thematic (for details, see Ruppenhofer et al., 2006). In FrameNet entries, each example is annotated with frame element labels. Let us consider an example:

(1) [<*protagonist*>She] didn't waitTgt [<*expected_event*>for a reply].

In this example, the verb *wait* is the frame evoker (marked as Tgt) and evokes the Waiting frame. The core frame elements of this frame are *protagonist* and *expected_event*, and "she" and "for a reply" are respectively assigned these labels. Thus, one of the most important assumptions is that at least one frame should be evoked in a sentence.

Another important aspect is that semantic frames are related to each other. There are several types of relations such as *inheritance*, *subframe*, and *use*, which indicate the hierarchical relationships of frames (for details, again, see Ruppenhofer et al., 2006). These relationships are visualized by the system called FrameGrapher. The following figure serves as an example:

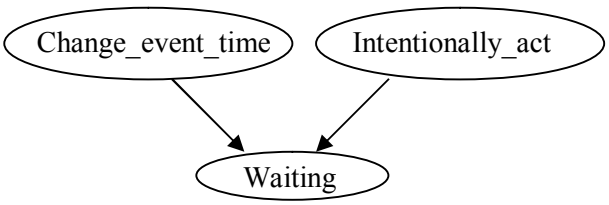


Figure 1. A FrameGrapher description of frame-to-frame relationships.

The descriptions of frame-to-frame relationships enable us to group specific frames into general categories. For example, the Waiting frame *uses*[4] the Intentionally_act frame, and we may say that the Waiting frame denotes a certain kind of action.

[3] The Second Edition of the BNC; http://www.natcorp.ox.ac.uk/.

[4] Here and elsewhere, the verb *use* in italics is used to refer to the frame-to-frame relationship "use," as defined in FrameNet.

2.2 Connectives in FrameNet

How, then, are connectives described in FrameNet? Let us look at some examples with connectives taken from the annotated data of FrameNet:

(2) [<*interlocutors*>They] sat chattingTgt [<*depictive*>together] [<*time*>*while* Elizabeth waited for trade to pick up again]. (FN)

(3) [<*time*>*While* Vernon and Lily were serving breakfast] [<*self_mover*>she] sneakedTgt [<*source*>out] and hid the crucifix behind a pile of Mr. Harcourt's empty cardboard boxes in the backyard. (FN)

The evoked frames are Chatting in (2) and Self_motion in (3). In both examples, *while* clauses are treated as *time* elements, which are the non-core frame elements of the Chatting and Self_motion frames, respectively. However, the fact that the designated meanings are different in (2) and (3) should not be overlooked. In (2), the *while* clause expresses an event occurring parallel to the event described in the main clause. By contrast, the *while* clause in (3) designates the timing of the event expressed in the main clause. This difference can be supported by the aspectual difference that the frame in the main clause of (3) is punctual, whereas it is not in (2). This indicates that descriptions of connectives in the present FrameNet annotation are insufficient for differentiating between the multiple senses of *while* in temporal uses.

In addition to the treatment of conjoined clauses as a non-core frame element, there are some clauses that are not assigned any semantic labels, such as *because* and *although* clauses. In the future edition, however, these clauses will be treated as extra-thematic elements (which are defined as peripheral parts of a sentence), as illustrated by the following example:

(4) Many teachers favor charter schools [<*concessives*>*although* their unions don't]. (Ruppenhofer et al., 2006:154)

The *concessives* category includes items such as *although*, *nevertheless*, and *despite*, all of which are assumed to express a concessive relation. The clauses headed by these items are in fact extra-thematic parts of the main clause. However, since the category includes various items, the syntactic and semantic differences among them cannot be captured sufficiently. Consider the following example:

(5) *Although/*Nevertheless* no longer a minister, Smithers still officiated at weddings.

The above example describes the situation where *although* and *nevertheless*—both of which are included in the *concessives*—cannot be used interchangeably. Thus, if we treat conjoined clauses as extra-thematic elements, we cannot differentiate between synonymous connectives such as *although* and *nevertheless*.

3 An extended FrameNet approach

Keeping in mind the issues discussed in Section 2.2, Section 3 proposes an extended FrameNet approach to connectives. There are three points to be noted: First, we consider not only the frame that is evoked in the main clause but also the frame that is evoked in the subordinate clause. Let us consider (2) again:

(2) [<*interlocutors*>They] sat chattingTgt [<*depictive*>together] [<*time*>*while* Elizabeth waited for trade to pick up again].

It should be noted that the *while* clause also contains a frame evoker, i.e., the verb *wait*. As noted above, this verb evokes the Waiting frame and, thus, (2) can be re-analyzed as follows:

(2') [<*interlocutors*>They] sat chattingTgt1 [<*depictive*>together] *while* [<*protagonist*> Elizabeth] waitedTgt2 [<*expected_event*>for trade to pick up again].

The second point in this analysis is the examination of the relationships between the frames in the main clauses (Fm) and those in the subordinate clauses (Fs). For the above example, we can formulate the relationship as (6):

(6) [Fm = Chatting; Fs = Waiting]

This is what we designate as a **frame valence**, which can concretely describe the types of semantic information realized in the clauses.

The third point is that we consider the general types of frames to seek generalization over specific frame valences such as (6). Frames will be categorized into the following six general categories: Activity, Attribute, Change, Location, Mental Attitude, and State. These types are mainly determined by frame-to-frame relationships. Activity frames are directly or indirectly related to the frames of Intentionally_act, Intentionally_affect, Activity, Motion, or Communication. For the first general type, Activity, aspectual properties are also considered and added to the specifi-

cation of the general type of frame with the coding <+pun> or <–pun>. Attribute frames indicate attributions of things or persons, such as price, age, and so forth. They are usually evoked by adjectival frame evokers. Change frames typically have the frame element *final_state* or *final_value*, which indicates the result of the change from the *initial_state* and *initial_value*. Location frames are frames that are related to the Locative_relation frame or the Residence frame. Mental Attitude frames refer to frames that have a *cognizer* or *experiencer* element and are typically related to the Emotion frame or the Mental Activity frame. Finally, the State frames refer to frames that *inherit from*[5] the State frame (which can be found in the FrameNet entry) or subframes of this frame. In addition, this category includes some frames denoting several continued processes, such as the Sleep frame and frames related to the Possession frame.

One of the advantages of using FrameNet data lies in the fact that frame-to-frame relationships are useful for generalizing the types of frames. To consider the general type with the above example (2), the Chatting frame denotes an activity of communication and the Waiting frame *inherits from* the Intentionally_act frame (see figure 1), and since both frames represent durative activities, (6) can be rephrased as (7):

(7) Activity<–pun>(Fm)-Activity<–pun>(Fs)

We will term this level of combinational patterns of frames a "**generalized frame valence**."

4 *While* and frame valences

This Section considers *while* as an example and examines how the frame valence approach can describe its meanings. As illustrated in the following definitions taken from *OALD*7,[6] *while* has various meanings:

(8)
a. at the same time as something else is happening
b. during the time that something is happening
c. used to contrast two things
d. (used at the beginning of a sentence) although; despite the fact that

For the purpose of convenience, we will use these definitions and designate each meaning as (8-a) simultaneity, (8-b) duration, (8-c) contrast, and (8-d) concession.

To inductively clarify the frame valences of *while*, we examine 50 core data mainly taken from the FrameNet examples and dictionaries such as *COB*5,[7] *MED*,[8] and *OALD*7. The sources of these examples are identified in parentheses after each example. Essentially, the labels of the frame elements are assigned to the examples; however, for the sake of simplicity, they are omitted in some cases. Although the assignment of these labels was carried out carefully, based on the FrameNet principle, in some cases, we had to depend on intuition. The frame valence of each example is provided in the form of [Fm *Frame name*; Fs *Frame name*], whereas generalized frame valences are marked as *Frame type*(Fm)-*Frame type*(Fs).

As a result of the analysis presented in the following sections, 10 generalized frame valences of *while* are observed, as indicated in the following table:

Meanings	No.	Fm	Fs
Simultaneity	i	Activity<–pun>	Activity<–pun>
	ii	State	Activity<–pun>
	iii	Activity<–pun>	State
Duration	iv	Activity<+pun>	State
	v	Activity<+pun>	Location
	vi	Activity<+pun>	Activity<–pun>
Contrast	vii	Change	Change
	viii	Attribute	Attribute
Concession	ix	Mental Attitude	Mental Attitude
	x	State	Mental Attitude

Table 1. The generalized frame valences of *while*.

In the following subsections, we will discuss each meaning of *while*, in turn, and show how the frame valence approach will differentiate between the multiple functions of *while*. In addition, we will briefly look at how synonymous connectives can be demarcated by frame valences, using the concessive *while* and *although* as examples.

4.1 Simultaneity *while*

This subsection examines *while* in the cases in which it has a simultaneous meaning. The most

[5] Here and elsewhere, the verb *inherit* (*from*) in italics is used to refer to the frame-to-frame relationship "inheritance," as defined in FrameNet.
[6] *Oxford Advanced Learner's Dictionary*, Seventh Edition.
[7] *Collins COBUILD Advanced Learner's Dictionary*, Fifth Edition.
[8] *Macmillan English Dictionary for Advanced Learners*.

typical frame valence of this use of *while* is what we observed in Section 3. Example (2) is reproduced below:

(2) They sat chattingTgt1 together *while* Elizabeth waitedTgt2 for trade to pick up again.
[Fm = Chatting; Fs = Waiting]
(i) Activity<–pun>(Fm)-Activity<–pun>(Fs)

Thus, we may state that the simultaneity *while* takes frames related to Activity both in Fm and Fs. (9) and (10) are examples of this type:

(9) [<*perceptive_agent*>They] listenedTgt1, doubts to the fore, *while* [<*speaker*>I] explainedTgt2 [<*topic*>the rudiments]. (*COB*5)
[Fm = Perception_active; Fs = Statement]
(10) [<*protagonist*>I waitedTgt1 *while* [<*agent*>he] gatheredTgt2 up [<*individuals*>his papers]. (*G*[9])
[Fm = Waiting; Fs = Gathering_up]

Example (9) contains the Perception_active frame in Fm and the Statement frame in Fs. The former frame contains *speaker* as a frame element and denotes an activity of communication; the latter *inherits from* the Intentionally_act frame. Similarly, (10) contains the Waiting frame, which *uses* the Intentionally_act frame in the main clause, and the Gathering_up frame—which *inherits from* the Intentionally_affect frame—in the subordinate clause. Hence, all the frames evoked in the clauses of (9) and (10) belong to the category of Activity.

In addition to this valence, there is another type that the following example illustrates:

(11) He used to bring along [<*agent*>his pet Alsatian], who satTgt1 patiently *while* [<*agent*>we] practisedTgt2 [<*action*>bandaging paws], she recalls. (FN)
[Fm = Posture; Fs = Practice]

In this example, the frame evoked in the main clause is Posture. This frame *inherits from* the State frame. As for Fs, the evoked frame is Practice, which *inherits from* the Intentionally_act frame and has a *duration* element. Thus, the frame valence in this example is generalized as (ii) State(Fm)-Activity<–pun>(Fs). In addition, we can hypothesize the reversed version of the valence, i.e., (iii) Activity<–pun>(Fm)-State(Fs), because Posture and Practice frames can also appear in the main and subordinate clauses, respectively, as follows: We practiced hard *while* he sat there.

4.2 Duration w*hile*

Let us now consider the durative use of *while*. First, consider the following example:

(12) *While* [<*helper*>Vernon and Lily] were servingTgt1 [<*focal_entity*>breakfast] [<*self_mover*> she] sneakedTgt2 [<*source*>out] and hid the crucifix behind a pile of Mr. Harcourt's empty cardboard boxes in the backyard. (FN)
[Fm = Self_motion; Fs = Assistance]

Both the frames evoked in the main and subordinate clauses, namely, Self_motion and Assistance frames, *inherit from* the Intentionally_act frame. If we consider the aspectual property of frames, one will note that the frame evoker *sneak* is punctual, and the Fm can be described as Activity<+pun>. By contrast, the frame in the subordinate clause has a durative nature and can be indicated as Activity<–pun>. To summarize, we propose that the generalized frame valence (iv) Activity<+pun>(Fm)-Activity<–pun>(Fs) is one of the basic schemata that provide the *while* sentence with the reading of duration.

Let us now consider some other types. The duration *while* also takes frames related to State and Location in subordinate clauses. The following are examples:

(13) I expect [<*self_mover*>the crew] had tiptoedTgt1 [<*path*>down] [<*manner*>very gently] [<*source*>from the bunk above] *while* [<*sleeper*> he] sleptTgt2. (FN)
[Fm = Self_ motion; Fs = Sleep]
(14) [<*communicator*>Someone] calledTgt1, *while* [<*person*>you] were outTgt2. (MED)
[Fm = Contacting; Fs = Expected_location_of_person]

In (13) and (14), all the frames evoked in the main clause denote some kind of punctual activity: Self_motion *inherits from* the Intentionally_act frame and the Contacting frame *uses* the Communication frame and describes the activity of making a phone call. Now considering the Fs, in (13), the Sleep frame is evoked by the adjective *asleep*. This frame denotes a continued process and hence, is a member of the State frame. With regard to (14), the Fs is the Expected_location_of_person. Since this frame *inherits from* the Locative_relation frame, it is a Location frame. These examples demonstrate that the duration *while* possesses the following two

[9] *Taishukan's Genius Unabridged English-Japanese Dictionary*.

frame valences: (v) Activity<+pun>(Fm)-State (Fs) and (vi) Activity<+pun>(Fm)-Location(Fs).

4.3 Contrastive *while*

Let us now turn to a discussion of the contrastive *while*. First, let us consider the following example:

(15) [<*group*>In Japan], [<*attribute*>sales] have plummetedTgt1 [<*difference*>20 percent] [<*time*> during that period], *while* [<*attribute*>European sales] have fallenTgt2 [<*difference*>12 percent].
[Fm = Change_position_on_a_scale;
Fs = Change_position_on_a_scale]

In this example, the same frame, i.e., Change_ position_on_a_scale, is evoked in both the main and subordinate clauses; this is one of the important characteristics of the frame valence patterns of the contrastive *while*. This fact implies that the contrastive meaning of *while* is realized not in the time domain but between frames, with a parallel relation in the main and subordinate clauses. (16) is another example of this use of *while*:

(16) [<*entity*>The south of the country] continues to growTgt [<*final_state*>richer], *while* [<*entity*>the north] growsTgt2 [<*final_state*>poorer]. (*MED*)
[Fm = Becoming; Fs = Becoming]

Both the main and subordinate clauses contain the Becoming frame, and it is clear that the same frame is also evoked in this example. If we consider the generalized frame valences of this *while*, we may note that the evoked frames in the above examples are frames related to Change: both Change_position_on_a_scale and Becoming frames have the FE *final_state*. Thus, we may state that one of the typical frame valences of the contrastive *while* is (vii) Change(Fm)-Change (Fs).

In addition to this valence, the following type of *while* is also observed:

(17) [<*goods*>The first two services] are freeTgt1, *while* [<*goods*>the third] costsTgt2 [<*asset*>£35]. (*COB*5)
[Fm = Expensiveness; Fs = Expensiveness]

The frame evoked in the main clause of example (17) is Expensiveness, and it is also evoked in the subordinate clause. Since this frame denotes an attribute of something, we can assume the following generalized frame valence: (viii) Attribute(Fm)-Attribute(Fs).

4.4 Concessive *while*

This subsection examines the concessive use of *while*. We first consider the frames evoked in subordinate clauses. The Fs of this usage tend to be the Mental Attitude frame. Consider the following examples:

(18) *While* [<*cognizer*>I] agreeTgt1 [<*cognizer*2> with you], [<*cognizer*>I] do not believeTgt2 [<*content*>that your way is best]. (MED)
[Fm = Certainty; Fs = Be_in_agreement_on_assessment]

(19) And *while* [<*experiencer*>I] likeTgt1 [<*content*>my job], [<*experiencer*>I] wouldn't want Tgt2 [<*event*>to do it] forever. (*CALD*[10])
[Fm = Desiring; Fs = Experiencer_subj]

(20) *While* [<*experiencer*>she] was very fondTgt1 [<*content*>of him], [<*experiencer*>she] didn't wantTgt2 [<*event*>to marry him]. (*YP*[11])
[Fm = Desiring; Fs = Experiencer_subj]

(21) *While* [<*cognizer*>I] am willingTgt1 [<*activity*>to help], [<*owner*>I] do not haveTgt2 [<*possession*>much time available]. (*OALD*7)
[Fm = Possession; Fs = Willingness]

The frames in the subordinate clauses are Be_in_ agreement_on_assessment in (18), Experiencer_ subj in (19) and (20), and Willingness in (21). These frames have either *experiencer* or *cognizer* elements, which typically mark individuals in frames who make judgments against something. These elements are also related to either Emotion or Mental Activity frames. This suggests that the frames evoked in the subordinate clauses of the concessive *while* are typically related to Mental Attitude.

Such a tendency is also observed in main clauses. In (18), the evoked frame in the main clause is Certainty, while in (19) and (20), it is Desiring. The Certainty frame contains a *cognizer* element and is indirectly related to the Mental Activity frame. The Desiring frame has an *experiencer* element and is related to the Emotion frame. Thus, most of the frames in the main clauses are also related to the Mental Attitude frame; hence, we can assume that one of the typical generalized valences of the concessive *while* is (ix) Mental Attitude(Fm)-Mental Attitude(Fs). In addition to this frame valence, the main clauses of the concessive use of *while* can also contain frames that are not related to Mental Atti-

[10] *Cambridge Advanced Learner's Dictionary*.
[11] *Shogakukan Youth Progressive English-Japanese Dictionary*.

tude such as Possession in (57). Since this frame belongs to the category of State, we can assume another frame valence, i.e., (x) State(Fm)-Mental Attitude(Fs). Overall, the patterns of the generalized frame valences of *while* can be summarized as seen in table 1.

4.5 Differentiating between *while* and *although*

Having demonstrated that the frame valence approach can successfully explain the polysemy of *while*, this subsection discusses how this approach differentiates between synonymous connectives, using *although* and the concessive *while* as examples. As is clear from the examples that will follow, although these items have similar meanings, they are not interchangeable in all cases:

(22)
a. *Although* he came and saved me, he hadn't heard me calling for help.
b. ?*While* he came and saved me, he hadn't heard me calling for help.
(23)
a. *Although* I was only six, I can remember seeing it on TV. (*COB*5)
b. ?*While* I was only six, I can remember seeing it on TV.
(24)
a. *Although* the tickets were expensive, the kids really enjoyed it.
b. ?*While* the tickets were expensive, the kids really enjoyed it.

We argue that these distributional differences can be accounted for by applying the notion of frame valence. We noted that the concessive *while* typically assumes frames that are related to Mental Attitude in the subordinate clause. If we examine the frames in the subordinate clauses of the above sentences, we find that they are Arriving (*come*), Age (*six*), and Expensiveness (*expensive*). The Arriving frame is related to Activity since it is a *subframe* of the Traversing frame that *inherits from* the Motion frame (this implies that it is indirectly related to the Motion fame), and Age and Expensiveness are frames of Attribute. With regard to the frames in the main clause, these are Hear in (22), Memory in (23), and Experiencer_subj in (24). The Hearing frame is a member of the Activity frame (since it *uses* the Communication frame), and the latter two frames are related to the Mental Attitude frame.

The unacceptability of the examples with *while* can be explained by the fact that frames related to Activity and Attribute are unlikely to appear in *while* clauses with concessive meaning, where mental attitudinal frames are expected to be evoked. On the other hand, these frames can appear in *although* clauses; hence, the examples with *although* are acceptable. Thus, it follows that the concessive *while* does not typically involve the frame valence Activity(Fm)-Activity (Fs) (= (22)), Mental Attitude(Fm)-Attribute(Fs) (= (23) and (24)). One possible explanation for this derives from the fact that Activity frames are typically evoked in the temporal use of *while* and Attribute frames are evoked in the contrastive meaning of *while*; in other words, they are usually not evoked in the clauses of the concessive *while*. Thus, the frame semantic analysis of connectives can successfully differentiate between the meanings of synonymous connectives.

5 Testing the frame valences of *while*

By examining a larger set of data (250 tokens) randomly drawn from the BNC, this Section deductively tests the list of the above obtained frame valences of *while*. For the sake of simplicity, we selected relatively short sentences in the written corpus and omitted sentences with frame evokers that do not appear in the FrameNet entry (e.g., *hesitate* and *look after*) and sentences with *while*, whose meaning is not simultaneity, duration, contrast, or concession (e.g., the conditional *while* as in the following: *While* the ravens are here the Tower will never fall). The results are summarized below:

Meanings	No.	Fm	Fs	Hit	Total
Simultaneity	i	Act<–pun>	Act<–pun>	39	69
	ii	State	Act<–pun>	14	
	iii	Act<–pun>	State	1	
Duration	iv	Act<+pun>	State	10	119
	v	Act<+pun>	Location	33	
	vi	Act<+pun>	Act<–pun>	59	
Contrast	vii	Change	Change	8	38
	viii	Attribute	Attribute	4	
Concession	ix	M_Attitude	M_Attitude	8	24
	x	State	M_Attitude	2	
Sum				178	250

Table 2. An examination of the frame valences of *while* in the BNC. Note: Act and M_Attitude denote Activity and Mental Attitude, respectively.

Out of 250 tokens, 178 matched the frame valence patterns that were obtained in Section 4 (71.2%). Regarding the other tokens that did not match our patterns, we can distill the following two points: 1) there are some frame valences other than the ones we proposed above; 2) it is possible to set up a better generalization of the frame valences of the contrastive *while*.

First, let us consider the durative use of *while*:

(25) Nothing happensTgt1 *while* you are rehearsingTgt2 in a garage.
[Fm = Event; Fs = Practice]

In the main clause of (25), the Event frame is evoked. Using a non-human subject, this frame does not designate a kind of activity; rather, it is just an event that may be generalized as Event (Fm). In the subordinate clause, the Practice frame, which *inherits from* the Intentionally_act frame, is evoked and is durative. Thus, we may generalize this valence as Event(Fm)-Activity<–pun>(Fs). It is also possible to assume Event(Fm) -Location(Fs) and Event(Fm)-State(Fs) based on the analogy between (iv) and (v). Indeed, in our present deductive analysis, we found one example for each valence. Another example of a newly found frame valence is Location(Fm)-Activity<–pun>(Fs) of the simultaneity *while*, as can be seen in the following examples:

(26) I'm aroundTgt1 *while* he doesTgt2 it.
[Fm = Locative_relation; Fs = Itentionally_act]

(27) I say I will useTgt1 it *while* I am hereTgt2 and give it back when I go.
[Fm = Using; Fs = Locative_relation]

Note that the latter case uses the reversed version of the valence: Activity<–pun>(Fm)-Location (Fs).

The second point is associated with the contrastive use of *while*, which yielded the lowest hitting ratio. This meaning of *while* can be better described by postulating that the contrastive meaning tends to be realized when the main and subordinate clauses evoke the same frame. Consider (28) and (29) below:

(28) In the North, clergy dominateTgt1 the catholic sector, *while* a protestant and loyalist culture predominatesTgt2 in the state schools…
[Fm = Dominate_situation; Fs = Dominate_situation]

(29) Of claimants who had a clinical examination, 90% were satisfiedTgt1 *while* 7% were dissatisfiedTgt2—mainly due to a perceived lack of thoroughness.
[Fm = Experiencer_obj; Fs = Experiencer_obj]

In (28), the Dominate_situation frame is evoked in both clauses. In terms of (29), if we consider the generalized type of the frames, it can be formulated as Mental Attitude(Fm)-Mental Attitude(Fs), which implies that the generalized valence closely resembles the frame valences of (ix) in the concessive meaning. However, the meaning of *while* in the sentence is more likely to be understood as contrast because the same frames are evoked in the main and subordinate clauses at the specific (not generalized) level of the frame valence. Hence, the parallel relations between the semantic information conjoined by the connective are preserved in the example. In addition, in both clauses, the *experiencer* elements are expressed in figures (i.e., 90% and 7%), which clearly indicates the contrast between the main and subordinate clauses. The fact that the same frame is evoked in both clauses in the contrastive use of *while* suggests that the frame valences can be reformulated as [Fm = Fs]. Furthermore, our deductive analysis revealed yet another characteristic of the contrastive *while*. Consider the following example:

(30) *While* 85 percent of telephone users are reportedly satisfiedTgt1 with the service they receive, commercial clients are not.

In this case, whereas the Experiencer_obj frame is evoked in the *while* clause, the predicate of the main clause is replaced with "not"; further, no frame evoker appears in this clause. It is safe to assume that the omitted predicate is "not satisfied," which again suggests that the same frame is evoked in both the clauses.

The test also revealed a weak point of the frame valences with regard to the concessive use of *while*. A reexamination of the sentences of the core data clarifies the fact that all the subjects in the sentences are animate, for example, "I" and "she." By contrast, (31) contains non-human subjects with *while*, which predominantly provides the following concessive reading:

(31) *While* the intentions of the plot are clear Tgt1 enough, it is not significantly developedTgt2 by the dance.
[Fm = Obviousness; Fs = Progress]

The subject in the clauses is "intentions." These are unlikely to be the element of the Mental Atti-

tudinal frame, metaphors being an exception. Thus, the frame valences of (ix) and (x) should be confined to the case wherein human subjects are involved. Other cases await more detailed investigation.

6 Conclusion

This paper has shown that the meanings of connectives can be described in a concrete manner by explicating the relations between the two frames evoked in both the main and subordinate clauses, namely, frame valences. We have presented (and tested) a list of generalized frame valences for describing the types of semantic entities conjoined with the connective *while*. This frame-valence-based approach is useful for differentiating between the polysemy of a connective and for distinguishing between synonymous connectives. In turn, this will improve the present system of FrameNet.

Thus, the position of this study sharply differs from those of previous studies. However, the view proposed here is firmly underpinned by the insights of previous works and, in turn, provides a better understanding of what has previously been stated about the descriptions of the meanings of connectives. In fact, our analysis of the meanings of *while* is based on the conceptualization of relations provided by the RST, and we do not intend to abandon the views proposed earlier.

Since FrameNet-based projects in other languages such as Japanese and Spanish are in progress, it is worth studying the meanings of connectives cross-linguistically. For instance, when comparing the results of our analysis on *while* with Japanese counterparts, we have observed that the simultaneity and duration uses of *while* may correspond to *aida* and *aida-ni*, respectively, in Japanese. In our preliminary analysis, there appears to be a strong tendency toward this correspondence. However, a more careful scrutiny, which might reveal subtle differences, needs to be conducted.

Though the list of frame valences must continue to be improved with a large-scale corpus study, we hope that the present paper has shed light on the meanings of *while* and offered insights into the ways of analyzing and describing connectives.

References

Baker, Collin F., Charles J. Fillmore and Beau Cronin. 2003. 'The Structure of the FrameNet Database.' *International Journal of Lexicography*, 16(3): 281-296.

Bednarek, Monika. A. 2005. 'Frames revisited: the coherence-inducing function of frames.' *Journal of Pragmatics* 37: 685-705.

Blakemore, Diane. 1987. *Semantic Constraints on Relevance*. Oxford: Blackwell.

Blakemore, Diane. 2000. 'Indicators and procedures: *nevertheless* and *but*.' *Journal of Linguistics* 36: 463-486.

Fillmore, Charles. J. 1982. 'Frame semantics.' In Yang, I. (eds.), *Linguistics in the Morning Calm: Selected Papers from SICOL-1981*. Seoul: Hanshin: 111-137.

Fillmore, Charles J., Christopher R. Johnson and Miriam R.L. Petruck. 2003. Background to Framenet, *International Journal of Lexicography*, 16(3): 235-250.

Fraser, Bruce. 1996. 'Pragmatic markers.' *Journal of Pragmatics* 6.2: 167-190.

Mann, William C. and Sandra A. Thompson. 1986. 'Rhetorical propositions in discourse.' *Discourse Processes* 9: 57-90.

Mann, William C. and Sandra A. Thompson. 1988. 'Rhetorical Structure Theory: Toward a functional theory of text organization.' *Text* 8 (3): 243-281.

Ruppenhofer, Joseph, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson and Jan Scheffczyk. 2006 'FrameNet II: Extended Theory and Practice.' Available online http://framenet.icsi.berkeley.edu/

Schiffrin, Deborah. 1987. *Discourse Markers*. Cambridge: Cambridge University Press.

Schourup, Lawrence. 1985. *Common Discourse Particles in English Conversation*. New York: Gerland.

Sweetser, Eve. 1990. *From Etymology to Pragmatics: Metaphorical and Cultural Aspects of Semantic Structure*. Cambridge: Cambridge University Press.

Dictionaries

CALD: *Cambridge Advanced Learner's Dictionary*. 2005. Cambridge: Cambridge University Press.

COB5: *Collins COBUILD Advanced Learner's Dictionary,* Fifth Edition. 2006. London: Collins.

G: *Taishukan's Genius Unabridged English-Japanese Dictionary*. 2001. Tokyo: Taishukan.

MED: *Macmillan English Dictionary for Advanced Learners*. 2002. Oxford: Macmillan Education.

OALD7: *Oxford Advanced Learner's Dictionary*, Seventh Edition. 2005. Oxford: Oxford University Press.

YP: Shogakukan Youth Progressive English-Japanese Dictionary. 2004. Tokyo: Shogakukan.

Resolution of Coordination Ellipses in Biological Named Entities Using Conditional Random Fields

Ekaterina Buyko
Jena University
Language & Information
Engineering (JULIE) Lab
Jena, Germany
buyko@coling-uni-jena.de

Katrin Tomanek
Jena University
Language & Information
Engineering (JULIE) Lab
Jena, Germany
tomanek@coling-uni-jena.de

Udo Hahn
Jena University
Language & Information
Engineering (JULIE) Lab
Jena, Germany
hahn@coling-uni-jena.de

Abstract

Coordination ellipsis is a linguistic phenomenon where relevant information is eliminated from the linguistic surface expression of a coordination. Thereby, the assumption is that readers can easily reconstruct the eliminated material given appropriate background knowledge. This phenomenon is particularly abundant in science jargon. As far as named entity recognition (NER) is concerned, failing resolution of coordination ellipsis leads to serious degradation in the performance of NER systems. Usually, they recognize elliptical coordinations, wrongly, as complex single entity mentions or classify correctly only non-elliptical parts of coordinations. As an alternative, we propose a methodology for decomposing complex coordinated entity expressions into constituent conjuncts, for determining the missing elements and thus reconstructing explicitly all of the single entity mentions. We present here a novel supervised machine learning-based approach to the resolution of elliptical coordinations in noun phrases. For the task of conjunct identification the model achieves performance of 93% on the biomedical GENIA corpus.

1 Introduction

Coordination constitutes one of the permanent 'hard' problems in linguistic theory and NLP for which no conclusive results have been worked out over the years. Part of the challenge is the high degree of ambiguity in syntactic analyses and the intricate interleaving between syntactic and semantic criteria for the proper interpretation of coordinated expressions.

Among those many crucial problems associated with coordinations, coordination ellipsis stands out as a linguistic phenomenon where relevant information is eliminated from the linguistic surface expression of a coordination on the assumption that readers can easily reconstruct the eliminated material given appropriate background knowledge. This phenomenon is particularly abundant in science jargon. Here, stunning implications of inappropriately dealing with coordinations can be observed for the proper recognition of named entities.

The problems we face can be illustrated by the following example taken from the biomedical domain we here focus on: *'the human c-Jun and JunB proteins'*. This expression poses several questions. Is *'human c-Jun'* conjoined with *'JunB proteins'* (the latter lacking the 'human' property), or is *'c-Jun'* conjoined with *'JunB'* (both sharing the 'human' property)? In essence, this noun phrase is a short form of two conjoined named entities *'the human c-Jun [proteins]'* and *'the [human] JunB proteins'*.

It should be fairly obvious that ignoring the mechanisms behind the recovery of deleted material (appearing in square brackets) by named entity taggers being used will inevitably result in the loss of biologically relevant entities. State-of-the-art named entity taggers typically recognize the phrase *'human c-Jun and JunB proteins'* as a single named entity (which is wrong because there are two entities involved) or, they might just recognize a part of this phrase *'JunB proteins'* as an entity (missing the other entity completely and also missing the *'human'* attribute for the recognized entity). We show here that elliptical coor-

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 163–171,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

dinations occur quite frequently in biological documents (see Section 4.2). Hence, the illustrated loss considerably impacts the quality of named entity recognition. In order to handle correctly such entity expressions, we propose a novel machine learning-based approach to the resolution of elliptical coordinations.

2 Elliptical Coordinations

Let us sort out the principal components of elliptical coordinations using the following examples of what we call here *elliptical entity expressions* (EEE):

1. *[alpha- and beta-]globin*
2. *[erythroid, myeloid and lymphoid] cell types*
3. *exons [2, 5, 7]*
4. *human [genes and cells]*
5. *[human and mouse] cells*
6. *[IL-5 / GM-CSF] receptor*

Consider, e.g., the phrase *'human and mouse cells'*, a simple case of coordination. In this example two entities *'human cells'* and *'mouse cells'* are semantically coordinated by the conjunction *'and'*. The occurrence of the conjunction allows to share some elements and to literally omit the token *'cells'* from the first part of the coordination. The result of the omission is an ellipsis (*'human ε and mouse cells'*) as indicated by 'ε'. In the second part of the coordination we find the shared element which is the ellipsis antecedent (*'cells'*). The omission of elements within a coordination results in an EEE. Each EEE contains at least two entity mentions and one of them can be handled literally as a discontinuous one (see Section 4). In our example, *'human and mouse cells'*, the discontinuous entity is *'human cells'*. In order to deal properly with EEEs, we need to resolve the elliptical coordinations (i.e., in the previous example we reconstruct the omitted element *'cells'* associated with *'human'*).

Generally speaking, the constituent parts of an elliptical coordination are conjuncts, conjunctions and shared elements (the so-called *ellipsis antecedent*).

Conjunctions – The coordinations can have some overt linker as in (1) or lack any (asyndetic coordinations) as in (3). As overt conjunctions we count conjunctive *'and'*, disjunctive *'or'*, and adversive *'but'*. Enumerations are often realized by the combination of conjunctions and the lack of these standard coordination signals as in (2). In these cases of asyndetic conjunctions the comma is often used to mark a missing conjunctive linker. In the biomedical domain, we also find a special linker, the slash, which may be used as a conjunctive linker as in (6).

Conjuncts – Minimally, a coordinated structure is binary (1), i.e., it contains two conjuncts. In most cases conjuncts are of the same syntactic category (nominal phrases (NP) in (4)). The conjuncts can be distinguished dependent on their position relative to the conjunction — *pre-conjuncts* precede the conjunction, while *post-conjuncts* follow the conjunction. The conjuncts are typically elements not only of the same basic category but of the same level of projection as well. They are either arguments (5) or heads (4). Concerning elliptical coordinations we introduce the notion of a *resolved conjunct*. This kind of conjunct is the result of the reconstruction from the original elliptical coordination by sharing the material surrounding the original conjuncts. For (4) we can identify two resolved conjuncts, *viz. 'human genes'* and *'human cells'*.

Ellipsis Antecedent – In our examples (see below), the ellipses are indicated by 'ε' and the correspondent shared material in the coordination (the so-called *antecedent* of the ellipsis) is enclosed in square brackets. The ellipses can be either forward ellipses (*analipsis*) (i.e., the ellipsis occurs in the pre-conjunct), or backward ellipses (*catalipsis*) (i.e., the ellipsis occurs in the post-conjunct). The combination of analipses and catalipses, as in the third example below, is a frequent case as well.

Analipsis *HL-60* ε_1 *and NB-4 [cells]*$_1$

Catalipsis *[protein phosphotase]*$_1$ *1 and* ε_1 *2A*

Combination of Analipsis & Catalipsis *[murine]*$_1$ *T* ε_2 *and* ε_1 *B [cells]*$_2$

The resolution of an elliptical coordination consists of two subprocesses, *viz.* the identification of the conjuncts followed by the resolution of the ellipsis, i.e., its binding to the elliptical antecedent proper. For *'murine T and B cells'* we have to identify the conjunction *'and'*, the conjuncts *'T'*, *'B'* and two ellipsis antecedents *'murine'* and *'cells'* which should be bound to corresponding positions of ellipses as in *'[murine]*$_1$ *T* ε_2 *and* ε_1 *B [cells]*$_2$*'*. The outcome are two resolved conjuncts, *viz. 'murine T cells'* and *'murine B cells'*.

3 Related Research

Coordination resolution has been addressed before by different methodological approaches. Statistical approaches enumerate all candidate conjuncts and calculate the respective likelihood according to a distribution estimated on a background corpus. For the coordination *'human cells and genes'* the distribution similarity of the candidate conjuncts *'genes'* and *'cells'* and the collocation frequencies of the pairs *'human'* - *'cells'* and *'human'* - *'genes'* would be compared against each other (cf., e.g., Chantree et al. (2005) for such an approach). However, only an f-measure under 50% was reported. Another purely statistical approach addressing only coordinations with PP attachments was presented by Goldberg (1999).

Other approaches rely on 'the principle of parallelism' (Beaugrande and Dressler, 1981). This principle considers those elements of a coordination as conjuncts which are most 'similar' under syntactic, morphological, or even semantic aspects. For instance, Agarwal and Boggess (1992) include syntactic and a lot of semantic information about candidate conjuncts and achieve an accuracy of about 82%. Okumura and Muraki (1994) estimate the similarity of candidate conjuncts by means of a similarity function which incorporates syntactic, orthographical, and semantic information about the conjuncts. The approach presented by Resnik (1999) is restricted to coordinations of the form *'(noun0) noun1 and noun2 noun3'*. The semantic similarity between putative conjuncts is calculated using WORDNET (Fellbaum, 1998). Resnik states that semantic similarity achieves higher coverage in coordination resolution than similarity measures based on morpho-syntactic information only. Using unsupervised web-based algorithms Nakov and Hearst (2005) achieved 80% on the disambiguation of coordinations of the fixed form *'noun1 and noun2 noun3'*.

The use of patterns which crucially rely on part-of-speech information, e.g., *'*N_2 *A C P* N_4*'*,[1] was proposed by Klavans and Jacquemin (1997). If such a pattern is matched by a coordination, [*A*] and [*P* N_4] are taken to be the conjuncts involved. The shortcoming of such an approach, however, is that many false positives are produced when the patterns are too general. If the patterns are overly specific, unfortunately, their number tends to explode.

The above mentioned methods focus on coordination resolution, in the general case. In this paper, however, we focus on the subproblem of elliptical coordinations in entity mentions only. Methods for named entity (NE) recognition[2] typically, either recognize NE coordinations as one single complex entity (especially, machine learning-based approaches) or only recognize non-elliptical parts of NE coordinations (dictionary-based approaches). Finkel et al. (2005) stipulate that the correct analysis and resolution of NE coordinations may improve results of NER. Mazur and Dale (2006) show the significance of the phenemenon of coordinated named entities in the newspaper domain, and present machine-learning classifiers to *disambiguate* the different uses of conjunctions. This work does not explicitly treat the *resolution* of elliptical coordinations, and thus cannot be directly compared with our approach.

To the best of our knowledge, there is only one alternative approach to the resolution of coordinated NEs in the biomedical domain by Nenadic et al. (2005). They propose patterns similar to the ones presented by Klavans and Jacquemin (1997) which are automatically generated from the GENIA corpus (Ohta et al., 2002). They conclude that patterns based on morpho-syntactic information are, however, insufficient – especially in the biomedical domain – and assume that more background knowledge would be necessary to avoid large amounts of false positives.

In order to address the problem of elliptical coordinations in entity mentions we, first, experi-

[1] This pattern represents expressions in French such as *'cycle cellulaire ou de developpement'*

[2] There is a huge body of work on this kind of named entity recognition, both on the general-language newspaper domain, as well for specific subdomains. Many approaches are discussed in the context of shared tasks, such as the Message Understanding Conferences MUC-6 (1995), the CoNLL shared task (Tjong Kim Sang and De Meulder (2003))), BioCreAtIvE (Hirschman et al. (2005)).

mented with a baseline inspired by the principle of parallelism. A novel approach based on (supervised) machine learning overcomes the inherent limitations of this baseline. This approach does not need any knowledge resources (as, e.g., Resnik's does) and yields, in contrast to the statistical approaches, high accuracy rates. We also provide a comprehensive evaluation on GENIA, a well-known corpus for the biomedical domain.

4 EEEs in Biomedical Corpora

4.1 Annotations

In the biomedical domain, there are basically two corpora with annotations of EEEs, *viz.* GENIA and PENNBIOIE.[3] Different annotation styles have been used in both corpora (Kim and Tsujii, 2006). Furthermore, they were assembled from rather specialized bio-subdomains. While GENIA deals with entities involved in biological reactions concerning transcription factors in human blood cells, the PENNBIOIE Oncology corpus deals with the molecular genetics of oncology.

The compositional approach of the GENIA corpus can be demonstrated looking at our example expression *'human and mouse gene'*. First, the lexical fragments of an expression are marked as constituents (*cons*). GENIA uses linguistic templates to describe the structure of the expressions. The constituents *'human'* and *'mouse'* are coordinated by the template *'X AND Y'* and compose the meaning '(AND 'human' 'mouse')'. The fragment *'gene'* is concatenated with the coordinated group composing the meaning '(AND 'human gene' 'mouse gene')'. The two composed expressions are then identified as a DNA family or group and yield 'G#DNA_family_or_group G#DNA_family_or_group' (see Figure 1 for the respective XML representation).

```
<cons lex="(AND human_gene mouse_gene)" sem="(AND G#
    DNA_family_or_group G#DNA_family_or_group)">
  <cons lex="human*">human</cons>
  and <cons lex="mouse*">mouse</cons>
  <cons lex="*gene">gene</cons>
</cons>
```

Figure 1: Annotation of EEEs in the GENIA corpus

The PENNBIOIE corpus employs a chaining approach to annotate discontinuous entity mentions in EEEs. Figure 2 shows the XML representation for the two discontinuous entity mentions *'codons 13'* and *'codons 61'* in the expression *'codons 12, 13 and 61'*. In the annotation, each fragment is marked as an *Annotation* element with a unique identifier (identifier *122* for *'codons'*, *123* and *125* for *'13'* and *'61'*). These are then linked by a *Chain* element.

```
<Annotation type="variation-location" span="560..566
    " ... id="122"></Annotation>
...
<Annotation type="variation-location" span="571..573
    " ... id="125"></Annotation>
...
<Annotation type="varlocation-location" span="
    571..573" ... id="127"></Annotation>
...
<Chain ids="122,125"/>
<Chain ids="122,127"/>
```

Figure 2: Annotation of EEEs in the PENNBIOIE corpus

4.2 Statistics about EEE in Biomedical Corpora

For both corpora, we have counted the number of elliptical entity expressions, the frequency of occurrence of the different conjunctions,[4] and the number of conjuncts in such an EEE. The number of all entity mentions involved in EEEs is about 5% in the PENNBIOIE Oncology corpus and around 3.5% in the GENIA corpus (see Table 1). The most frequent conjunctions are *'and'*, *'/'*, *','* and *'or'*. In both corpora, most elliptical entity expressions contain two conjuncts; the maximum number of conjuncts found in these corpora is seven. As shown before by Nenadic et al. (2005), 90% of the conjuncts are arguments of the coordinated NP. That means that analipses predominate in the biomedical corpora.

	GENIA		PENNBIOIE ONC.	
All NEs	97876	100%	7677	100%
NEs in EEEs	3426	3.5%	400	5%
EEEs	1578		172	
Conjunction	*'and'*	74%	'/'	35%
	'or'	8%	*'and'*	32%
	', , and'	4%	','	15%
	other	14%	other	18%
Number of conjuncts	2	90%	2	66%
	3	4%	3	21%
	> 3	6%	> 3	13%

Table 1: EEEs in the GENIA and PENNBIOIE Oncology corpora

[3] http://bioie.ldc.upenn.edu

[4] Both, GENIA and PENNBIOIE consider the following conjunctions: 'and', ',' , 'or', 'both ... and', 'either ... or', 'but not', 'neither ... nor', and '/'.

5 Methods

Our approach to EEE resolution consists of two steps, *viz.* the recognition of entity mentions (including candidate EEEs), followed by the resolution of EEEs, where the second step is divided into two substeps: first, the identification of the conjuncts, and second, the resolution of the ellipses (see Figure 3). Throughout this section, we will refer to the following example: *'IL-2 and IL-4 receptors'*. In this EEE, we aim to identify two entity mentions: *'IL-2 receptor'* and *'IL-4 receptor'*.

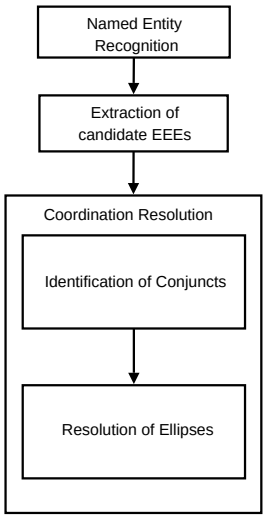


Figure 3: Resolution of Elliptical Entity Expressions (EEEs)

5.1 Recognition of Entity Mentions

In a first step, we recognize and classify entity mentions in text. Our approach to entity recognition is based on the one presented by Settles (2004) who uses Conditional Random Fields (CRFs; Lafferty et al. (2001)), i.e., probabilistic conditional models for relational learning (see also Section 5.2.2), and employs a rich set of binary features. We have slightly modified this approach by extending the feature set (e.g., adding POS features and considering the stemmed form of the tokens only). EEEs are considered as single entity mentions. *'IL-2 and IL-4 receptors'*, e.g., would thus be predicted as one entity. Non-elliptical coordinations such as *'T cells and B cells'* are by definition not EEEs and should immediatly be predicted to contain two distinct entity mentions *'T cells'* and *'B cells'*.

Entity mentions recognized by our NER are marked to be candidate EEE if they contain at least one of the elements of the set of conjunctions (see Table 1). The candidate EEEs can be divided into two groups. The first group contains expressions which are positive EEEs with at least two entities (*'human and mouse genes'*). The second group of false positives contains expressions with the conjunction as part of the entity (*'signal transducer and activator of transcription (STAT) protein'*. The latter group is only marginal in biomedical EEEs. Therefore, we ignore this distinction and proceed with all candidate EEEs.

5.2 Identification of Conjuncts

In this step, we identify conjuncts within the extracted candidate EEEs. Given our example *'IL-2 and IL-4 receptor'* we should then identify two conjuncts *'IL-2'* and *'IL-4'*. We here present two approaches to conjunct identification, *viz.* pattern-driven baselines, and a machine learning-based approach. Whereas many algorithms are restricted to binary coordinations, both of our approaches can handle *n*-ary coordinations.

5.2.1 Baselines

In the first simple baseline (Baseline 1) we assume that the last token in the candidate EEE is a shared element. This implies that tokens neither marked as shared elements nor as conjunctions are conjuncts.

Our second baseline (Baseline 2) builds on 'the principle of parallelism'. We focus on syntactic symmetry of conjuncts constituted by their respective part-of-speech (POS) information.

The input for the baseline is a sequence $\mathbf{o} = o_1, ..., o_n$, where o_i is a token at position i, $0 < i \leq n$. For a sequence with m conjunctions, the algorithm should identify $m+1$ conjuncts, i.e., for each conjunction at position o_k in the sequence, the following conjuncts have to be identified: the pre-conjunct $c_{pre} = o_i, ..., o_{k-1}$, $(0 < i < k)$ and the post-conjunct $c_{post} = o_{k+1}, ..., o_i$, $(k < i \leq n)$.

The identification of conjuncts is based on a word-by-word comparison of token POS information. For each conjunction at position o_k the algorithm constructs the pre- and post-conjuncts by iteratively matching the POS tags of o_{k-d} and o_{k+d}, where o_{k-1} is the token left of the conjunction. The algorithm stops when the POS tags of o_{k-d} and o_{k+d} differ. Thus, the algorithm creates the maximal symmetry of conjuncts with respect to the POS tags.

Restrictions of the baseline are the direct propagation of errors from the POS tagger, and the word-by-word comparison which results in the comparison of word sequences only with the same length rather than, e.g., complex grammatical categories at the phrasal level.

5.2.2 Supervised Approach

In order to battle the restrictions of our baseline, we propose here a new approach for the coordination resolution based on supervised ML. The goal is here to construct a statistical model from a set of training instances, which can then be used to identify conjuncts in coordinations. Such a coordination consists of a sequence of single tokens, for each of which we wish to assign a label sepending on its function in the elliptical coordination: *'C'* for the conjuncts, *'CC'* for the conjunctions, and *'A'* for the ellipsis antecedent.

The task at hand can thus be considered similar to other chunking tasks where a sequence of linguistic units – tokens in a sentence, mostly – needs to be assigned labels. This could be handled as a traditional classification task where all units of a sequence are considered mutually independent[5] and their labels are thus predicted separately. If, however, there is an inherent interdependency between the single units in a sequence and the independence assumption does not hold, this may lead to a suboptimal classification performance. This is the case in many NLP chunking tasks, including POS tagging, shallow parsing, and especially entity recognition. Under the presence of such statistical relational data, the application of sequence models should be favored, which assign a sequence of labels to the complete input sequence of units.

Conditional Random Fields (CRF) are probabilistic sequence models which have recently been shown to be well suited for such relational decision problems, e.g., different linguistic chunking tasks (cf. Settles (2004) and Sha and Pereira (2003)). A CRF can be interpreted as an undirected graphical model which is globally conditioned on the input observation sequence o and trained so that it maximizes the conditional probability of the label sequence $\mathbf{l} = l_1, ..., l_n$ and the observation sequence $\mathbf{o} = o_1, ..., o_n$. A CRF can be arbitrarily structured; however, in most NLP applications, the underlying graph is a simple linear chain. In the following, we will only focus on this case. Conditional probability is thus defined as:

$$P(\mathbf{l}|\mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp(\sum_j \sum_i \lambda_j f_j(l_{i-1}, l_i, \mathbf{o}, i))$$

where $Z(o)$ is a normalization factor over all label sequences. A feature function f_j relates the current observation with a bigram of labels; i.e., it is an indicator function associating an observation feature at position i (e.g., *word at current position i is 'and'*) and a particular state transition (e.g., *state at position* $i-1$ *is 'C' and state at current position i* is 'CC'). The feature weights λ_j are estimated by maximizing the respective conditional log-likelihood function on the training data by means of iterative-scaling or gradient-based techniques. The higher the value of λ_j, the more important this feature function is.

For our approach to ML-based conjunct identification, we use the CRF implementation available in the machine learning toolsuite MALLET (McCallum (2002)). We employ a rich set of binary features, including orthographical, lexical, morphological, syntactic and context features (see Table 2). Furthermore, we model the output labels of the Baseline 2 as features.

Feature Class	Description
lexical and morphological	stemmed version of each token, prefix and suffix of length 3
orthographical	based on regular expressions (e.g. *HasDash*, *IsInitCaps*, *Punctuation* ...), token transformation rule: capital letters replaced by 'A', lowercase letters by 'a', etc. (e.g. *IL2* → *AA0*, *have* → *aaaa*)
morpho-syntactic	the token's part-of-speech tag
contextual	conjunctions of all features of neighboring tokens (two tokens to the left and one token to the right)
token position	last token in the sequence
baseline 2 output	output labels of the baseline 2 (e.g., *'C'*, *'CC'* and *'A'*)

Table 2: Feature classes used for conjunct identification

5.3 Resolution of Ellipses

Having identified the conjuncts of a coordination, a rule-based heuristic is employed to fill the ellipses in order to reconstruct the single entity mentions of an EEE. An essential property of a coordination is that conjuncts share the same ellipsis antecedent. To reconstruct all entity mentions in an

[5]Formally, this is based on the assumption that all these units are independent and identically-distributed examples.

elliptical coordination we have to associate each conjunct with the shared elements.

We may assume that there are shared elements (ellipsis antecedent) if there are tokens which are neither marked as conjuncts nor as conjunctions. To construct the resolved conjuncts from the identified conjuncts and the shared elements, we first mark the positions of shared elements and identify then the positions of their ellipses, i.e., whether the shared element should be copied before or after the conjunct. Second, we have to correct the number form of the shared element. In our example, we would identify *'receptors'* as the shared element and the ellipsis position ε: *'IL-2 ε and IL-4 receptors'*. We could then create the resolved conjuncts by binding the shared element *'receptors'* to the ε and normalizing the plural form. The final result of the EEE resolution are two resolved conjuncts *'IL-2 receptor'* and *'IL-4 receptor'* which are coordinated by the conjunction *'and'*.

6 Results and Discussion

6.1 Data Set

For our evaluation we extracted all EEEs from the GENIA corpus where we found, in total, 1578 EEEs (1442 distinct EEEs) (see Table 1). For training and evaluation purposes, the extracted EEEs were converted into a representation of tokens labeled as *'C'* for conjunct, *'CC'* for conjunction, and *'A'* for the ellipsis antecedent. In order to evaluate the task of conjunct identification, we used recall and precision scores since they are well suited for the evaluation of segmentation tasks. Thereby we evaluated the assignment of *'C'* labels denoting conjuncts. We have carried out the tenfold cross-validation of methods for the conjunct identification (cf. Table 3).

In order to evaluate the overall task of EEE resolution (conjunct identification and resolution of ellipses), we have randomly split the data into training data (90%, for the ML-approach) and evaluation data (10%, used for the evaluation of all approaches). Thereby we evaluated the accuracy of EEE resolution as the accuracy of coordination resolution. The coordination is resolved, if all conjuncts in the coordination are correctly identified and all shared elements are correctly bound. The evaluation results are summarized in the Table 3 and Table 4.

	Recall	Precision	F-measure
Baseline 1	68%	70%	69%
Baseline 2	54%	85%	66%
CRF	**93%**	**93%**	**93%**

Table 3: Conjunct identification in EEEs, cross-validation on the GENIA corpus

	Accuracy
Baseline 1 + heuristic	51%
Baseline 2 + heuristic	63%
CRF + heuristic	**86%**

Table 4: EEE resolution, evaluated on the GENIA corpus (158 randomly selected EEEs)

6.2 Baselines

The first simple baseline achieves an f-measure of 69%. The evident shortcoming of this baseline is that it cannot correctly treat catalipses such as *'exons 1, 2, 3'* and combinations of catalipses and analipses such as *'human T and B cells'*. With the conjunct identification, our second baseline achieves an f-measure of 66%. A high precision of 85% indicates that the POS information is a reliable information source to identify symmetry in the coordination. Still, the limitations of a baseline dealing only with POS tags are reflected by a rather low recall of 54%. There are two typical error types. First, coordinations, such as *'erythroid_JJ and_CC T-cell_NN lines_NNS'* cannot be resolved properly due to different POS tags of the conjuncts. Second, when coordinations have conjuncts of different lengths (*'NF-kappa B and STAT-6 binding sites'*), the baseline also fails. Together with our heuristic for ellipsis interpretation Baseline 1 and Baseline 2 yield accuracies of 51% and 63% on the overall task of EEE resolution respectively (see Table 4).

6.3 Supervised Approach

Our ML approach outperforms both baselines significantly and yields an f-measure of 93% (see Table 3). The higher performance of the ML-based approach as compared to the baseline is mainly due to the fact that the ML-based approach properly handles coordinations where the conjuncts have different lengths and is more robust with respect to different POS tags. We measured the degrees of feature contribution for this task (see Table 5).

The results show that the lexical and morphological features only contribute to the performance of 89.5%. Within other feature groups, the con-

	Recall	Precision	F-measure
lexical, morphological	89.7%	89.2%	89.5%
+ orthographical	90.3%	89.6%	89.9%
+ morpho-syntactic	90.9%	90.3%	90.6%
+ contextual	92.1%	91.4%	91.8%
+ token position	92.3%	91.7%	92.0%
+ baseline 2 output	**93.3%**	**92.7%**	**93.0%**

Table 5: Feature contribution for conjunct identification in EEEs with CRF, tenfold cross-validation on the GENIA corpus

textual features and the features of Baseline 2 enable performance increases of more as two percentage points. As with the task of EEE resolution, the ML-based approach in combination with the heuristic for ellipsis interpretation achieves an accuracy of 86% (see Table 4).

Analyzing the errors made by our ML approach, we identified three groups. First, annotation errors in the evaluation data. Second, very tricky expressions such as nested elliptical coordinations (*'recombinant human nm23 -H1, -H2, mouse nm23 -M1, and -M2 proteins'*). And third, most of the errors occur in EEEs containing both analipses and catalipses. For example in the EEE *'human T and B cells' 'human'* was erroneously identified as the part of the first conjunct and not as a shared element. This problem could be addressed by a richer feature set which should especially include more context information. Also, a more sophisticated CRF structure (instead of the linear chain) could probably more adequately represent the interdependencies between the elements in a coordination.

7 Conclusions

In this paper, we focused on elliptical coordinations of named entities in biomedical documents. Proper resolution of such elliptical entity expressions is an important step to avoid errors in NLP processing steps, such as NE recognition and relation detection. We propose a novel supervised machine learning approach to the identification of the conjuncts and an overall approach to the resolution of elliptical coordinations based on CRF. On the GENIA corpus, the approach to conjunct identification yields an f-measure of 93% and does significantly outperform pattern-driven baselines. For the overall task of EEE resolution our approach achieves an accuracy of 86%. Hence, CRF may be well suited for the resolution of elliptical NP coordinations.

Future work will investigate whether a differently structured CRF with improved features might be able to handle coordinations with both analipses and catalipses better than our current approach. Furthermore, we plan to evaluate a complete task of NER including our approach for the elliptical coordination resolution.

Acknowledgments. This research was funded by the ECs 6th Framework Programme (4th call) within the BOOTStrep project under grant FP6-028099, and by the German Ministry of Education and Research (BMBF) via its e-Science initiative within the StemNet project (funding code: 01DS001A to 1C).

References

Rajeev Agarwal and Lois Boggess. 1992. A simple but useful approach to conjunct identification. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 15–21. Newark, Delaware, USA, 1992. Association for Computational Linguistics.

Robert Beaugrande and Wolfgang Dressler. 1981. *Introduction to Text Linguistics*. Longman Group Limited, Harlow, Essex.

Francis Chantree, Adam Kilgarriff, Anne de Roeck, and Alistair Willis. 2005. Disambiguating coordinations using word distribution. In *RANLP 2005 – Proceedings of the International Conference on 'Recent Advances in Natural Language Processing'*, pages 144–151. Bulgaria, 2005.

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

Jenny Finkel, Shipra Dingare, Christopher D. Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. 2005. Exploring the boundaries: gene and protein identification in biomedical text. *BMC Bioinformatics*, 6(Supplement 1).

Miriam Goldberg. 1999. An unsupervised model for statistically determining coordinate phrase attachment. In *Proceedings of ACL '99, 37th Annual Meeting of the Association for Computational Linguistics*, pages 610–614. College Park, Maryland.

Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. 2005. Overview of BIOCREATIVE: Critical assessment of information extraction for biology. *BMC Bioinformatics*, 6(Supplement 1: S1).

Jin-Dong Kim and Jun'ichi Tsujii. 2006. Corpora and their annotation. In Sophia Ananiadou and John McNaught, editors, *Text Mining for Biology and Biomedicine*, pages 179–211. Norwood, MA: Artech House.

Judith Klavans and Christian Jacquemin. 1997. A natural language approach to multi-word term conflation. In *Proceedings, DELOS Workshop on Cross-Language Information Retrieval, ETHZ, Zurich.* ERCIM: European Consortium for Informatics and Mathematics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML-2001 – Proceedings of the 18th International Conference on Machine Learning*, pages 282–289. Williams College, MA, USA, June 28 - July 1, 2001. San Francisco, CA: Morgan Kaufmann.

Pawel Mazur and Robert Dale. 2006. Named entity extraction with conjunction disambiguation. In *LREC 2006*

– *Proceedings of the 5th International Conference on Language Resources and Evaluation.*, pages 1752–1755. Genoa, Italy, 24-26 May 2006. Paris: European Language Resources Association (ELRA).

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

MUC-6. 1995. *Proceedings of the 6th Message Understanding Conference*. Columbia, Maryland, November 6-8, 1995. San Mateo, CA: Morgan Kaufmann.

Preslav Nakov and Marti Hearst. 2005. Using the web as an implicit training set: Application to structural ambiguity resolution. In *HLT-EMNLP 2005 – Proceedings of EMNLP*, pages 835 – 842.

Goran Nenadic, Irena Spasic, and Sophia Ananiadou. 2005. Mining biomedical abstracts: What's in a term? In Keh-Yih Su et al., editor, *Natural Language Processing - IJCNLP 2004 First International Joint Conference , Lecture Notes in Computer Science*, volume 3248.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. In M. Marcus, editor, *HLT 2002*, pages 82–86. San Diego, Cal., USA, March 24-27, 2002. San Francisco, CA: Morgan Kaufmann.

Akitoshi Okumura and Kazunori Muraki. 1994. Symmetric pattern matching analysis for English coordinate structures. In *ANLP 1994*, pages 41–46. Stuttgart, Germany, 13-15 October 1994. San Francisco, CA: Morgan Kaufmann.

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.

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *JNLPBA*, pages 107–110. Geneva, Switzerland, August 28-29, 2004.

Fei Sha and Fernando C. N. Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL 2003*, pages 213–220. 2003, Edmonton, Canada. San Francisco, CA: Morgan Kaufmann.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *CoNLL-2003 – Proceedings of the 7th Conference on Computational Natural Language Learning*, pages 142–147. Edmonton, Canada, 2003. Association for Computational Linguistics.

Minimising semantic drift with Mutual Exclusion Bootstrapping

James R. Curran and **Tara Murphy** and **Bernhard Scholz**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{james,tm,scholz}@it.usyd.edu.au

Abstract

Iterative bootstrapping techniques are commonly used to extract lexical semantic resources from raw text. Their major weakness is that, without costly human intervention, the extracted terms (often rapidly) *drift* from the meaning of the original seed terms.

In this paper we propose *Mutual Exclusion bootstrapping* (MEB) in which multiple semantic classes compete for each extracted term. This significantly reduces the problem of semantic drift by providing boundaries for the semantic classes. We demonstrate the superiority of MEB to standard bootstrapping in extracting named entities from the Google Web 1T 5-grams. Finally, we demonstrate that MEB is a *multi-way cut problem* over semantic classes, terms and contexts.

1 Introduction

Extracting lexical resources from text is a central problem in Natural Language Processing. These resources are the key to overcoming the knowledge bottleneck in tasks ranging from Word Sense Disambiguation to Question Answering.

Template-based approaches have been very successful – they can be implemented efficiently, work on small- and large-scale datasets, and require minimal linguistic pre-processing, so are largely language independent. Template-based relation extraction was pioneered by Hearst (1992), who demonstrate that hyponyms could be extracted using templates like X, ..., Y and/or other Z where X, ..., Y are hyponyms of Z. Berland and Charniak (1999) use a similar approach to identify whole-part relations and Caraballo (1999) uses the extracted hyponyms to build a hierarchy. The disadvantage of these fixed templates is that matches are very rare, resulting in low term recall.

Riloff and Shepherd (1997) propose *iterative bootstrapping* where related terms that are frequent neighbours to terms in the semantic class are extracted and Roark and Charniak (1998) improve accuracy by altering the bootstrapping parameters. In *mutual bootstrapping* (Riloff and Jones, 1999), both the terms and the contexts they occur in are extracted. Agichtein and Gravano (2000) and Agichtein et al. (2000) use similar approaches for Information Extraction (IE), such as identifying company headquarters, and Sundaresan and Yi (2000) identify acronyms and their expansions.

Bootstrapping has the advantage that it can identify new templates or *contexts*, which in turn can identify new terms, significantly increasing recall. Unfortunately, adding only a term with a different predominant sense, or a context that weakly constrain the terms, can quickly introduce errors. Therefore, a common theme in the evaluation of bootstrapping is *semantic drift*, when these erroneous terms or contexts infect the semantic class.

We propose a new stricter form of bootstrapping, *Mutual Exclusion Bootstrapping* (MEB), which minimises semantic drift using *mutual exclusion* between semantic classes. Each class is extracted in parallel using separate bootstrapping instances that compete to extract terms and contexts. We add *stop classes* that collect terms known to cause drift in particular semantic classes.

We compare MEB against mutual bootstrapping for extracting BBN named-entity types (Weischedel and Brunstein, 2005) from the 5-grams of the Google Web 1T corpus. We demonstrate that MEB outperforms mutual bootstrapping, can scale to massive datasets, and works well on noisy web text. We also evaluate distributional similarity approaches on this dataset, finding that bootstrapping is faster and more accurate.

Finally, we show that the MEB algorithm is an instance of *multi-way cut*, the generalisation of the

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 172–180,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

min-cut graph problem. Although multi-way cut is NP-hard, we demonstrate the feasibility of using approximation algorithms to find near optimal partitions of contexts and terms into semantic classes.

2 Mutual and Multi-level Bootstrapping

Riloff and Jones (1999) have proposed *mutual bootstrapping* (MB), where both the terms, and the contexts used to extract terms, are extracted in alternating bootstrap iterations. First a small set of seed words are used to find possible contexts. These contexts are ranked according to

$$\text{score}(c_i) = \frac{\text{seen}(c_i)}{\text{new}(c_i)} \log_2(\text{seen}(c_i)) \qquad (1)$$

where $\text{seen}(c)$ is the number of terms (by type) extracted with context c that are already in the semantic class, and $\text{new}(c)$ is the total number of terms (by type) extracted with context c. MB is designed to balance reliability and productiveness of the context. The highest scoring context is added to the semantic class. The terms that occur in the context are then added to the semantic class.

Riloff and Jones (1999) also introduce *multi-level bootstrapping* to overcome the problem of semantic drift. Rather than adding all of the extracted terms, multi-level bootstrapping only adds the five most reliable terms in each iteration. If a term is extracted by more contexts already in the semantic class then it is more reliable, with a small additional weighting for the score for each context.

We simplify the scoring functions in our implementation, making the scoring symmetrical for terms and contexts. The contexts are ordered by the number of terms in the semantic class they extract (*reliability*). Ties are broken by taking the context that would add the most new terms (*productivity*). In this way, the scoring function prefers precision over recall as much as possible.

Terms are ordered in the same way with respect to contexts. In each iteration a fixed number of contexts and then terms are added to the semantic class, thus we perform multi-level bootstrapping on both the terms and contexts.

3 Mutual Exclusion Bootstrapping

Mutual Exclusion Bootstrapping (MEB) attempts to minimise semantic drift in both the terms and contexts. It does this by extracting multiple semantic classes in parallel, using multiple independent bootstrapping instances, except that a term or context *must only be used by one* bootstrapping instance. We assume that the terms only have a single sense and that contexts only extract terms with a single sense, that is, the semantic classes are *mutually exclusive* with respect to terms and contexts.

in : Seed word lists S_k $\forall$ categories k
in : Raw contexts $\mathscr{C}$ and terms $\mathscr{T}$
in : # terms N_T and contexts N_C per iteration
out: Term T_k and context C_k lists $\forall$ category k
$T_k \longleftarrow S_k \forall$ categories k;
foreach *iteration* **do**
  foreach $c \in \mathscr{C}$ **do**
    count the number of times c occurs with $t \in T_k$;
    discard c if occurs with multiple classes;
  foreach *class* k **do**
    sort set of c by above occurrence counts;
    add top N_C contexts to C_k;
  foreach $t \in \mathscr{T}$ **do**
    count the number of times t occurs with $c \in C_k$;
    discard t if occurs with multiple classes;
  foreach *class* k **do**
    sort set of t by above occurrence counts;
    add top N_T terms to T_k;

Algorithm 1: Mutual Exclusion Bootstrapping

This assumption is far from correct, although for many terms including the named entities we consider here, there is a clearly dominant semantic class. For some pairs of semantic classes, e.g. nationalities and languages, have a significant lexical overlap and are far from mutually exclusive. Interestingly, we see the best results by *artificially forcing these categories apart*. As our experiments show, this enables us to distinguish classes which are quite hard to distinguish otherwise.

The MEB algorithm is shown in Algorithm 1. In each iteration, contexts and then terms are added to each semantic class. If more than one class attempts to extract a context or term then it is eliminated, leading to mutual exclusion between the semantic classes. The terms and contexts are scored and ordered in the same way as our mutual bootstrapping implementation – the only addition in MEB is the parallel mutual exclusion constraint.

The mutual exclusion is very strict and so a large number of terms and contexts are thrown away. This is not a major issue when we are using such a large dataset as the Web 1T corpus, but could be a more significant problem on smaller datasets. It is also more of a problem if there is sig-

nificant lexical overlap between semantic classes.

Notice that the algorithm is sensitive to the order in which contexts and terms are added to the semantic classes, since once they are added to a class they cannot be used elsewhere. For example, if a minority sense of a term is identified by a context first, it may be added to the minority class rather than dominant class for that term. This has the potential to cause drift in the same way as occurs in the original bootstrapping algorithms.

4 Using the Google Web 1T n-grams

Riloff and Jones (1999) used contexts extracted by AutoSlog-TS (Riloff, 1996) from text that had been shallow parsed to identify NPs, VPs and PPs. This means a POS tagger and chunker must be available in the target language, making their approach language dependent. In our experiments, we wanted to take a completely language independent approach where possible. We also wanted to demonstrate that MEB could scale efficiently to extremely large datasets, because these datasets provide the levels of redundancy needed to overcome the sparseness of the extracted contexts.

Google has recently released the Web 1T corpus (Brants and Franz, 2006), which consists of unigram to 5-gram counts calculated over 1 trillion words of web page text collected in January 2006. The text was tokenised following the Penn Treebank tokenisation, except that words are usually split on hyphens, and dates, email addresses and URLs are kept as single tokens. The sentence boundaries are marked with two special tokens `<S>` and `</S>`. The individual terms in the n-grams occurred at least 200 times otherwise they were replaced with the special token `<UNK>`. The n-grams themselves must appear at least 40 times to be included in the Web 1T corpus.

We use the 5-grams from the Web 1T corpus as our raw text, such that the middle token is the *term* and the two tokens on either side form the *context*. The advantage of this context definition is that it is quite language independent. The disadvantage is that we can only extract terms consisting of a single word and the contexts are noisier than those extracted from the shallow parsed text.

We filter out 5-grams in several ways. We remove all 5-grams where the middle token is not title case because we are only extracting proper noun named-entity types. We also remove all contexts that include numbers. Finally, we eliminate contexts that only appear with one term and thus terms that only appear with one context, since they cannot be reached by the bootstrapping algorithm. The size of the resulting dataset is shown in Table 1. We have reduced the 1 trillion n-grams down significantly with filtering, so we only using 2% of the data by type and 3.6% of the data by token. However, the number of terms and contexts by type is still extremely large. The dataset is 666MB on disk which all needs to be loaded into memory at once.

TYPE	COUNT
Number of terms	694 047
Number of contexts	10 597 784
Number of unique instances	42 807 058
Number of instances	21 308 744 742

Table 1: Filtered Web 1T dataset statistics.

5 Implementation

The MEB implementation has been optimised to be as time and space efficient as possible. Each unique term that appears with a context requires only 4 bytes of storage, which means the program requires around 1GB of RAM to run. The terms and contexts that co-occur are completely cross-indexed which makes updating the term and context extraction counts very efficient. Finally, the mutual exclusion property means that the term and context sets for each semantic class can be represented implicitly using flags, so the many set membership tests are also extremely fast. The bootstrapping experiments described here take only minutes to run and much of that time is spent loading the data into memory.

6 Selecting semantic classes

In these experiments, we wanted to extract semantic classes corresponding to proper-noun named entities only. We based our semantic classes on the 29 entity types used to annotate the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005) distributed by the LDC. The BBN corpus includes detailed entity annotation guidelines which helped with the evaluation process described below.

We ignored many entity types that did not primarily involve proper nouns, including DESCRIPTION types, CHEMICALS and SUBSTANCES, TIMES, MONETARY amounts and QUANTITIES

LABEL	DESCRIPTION
FEM	Person: female first name *Mary Patricia Linda Barbara Elizabeth*
MALE	Person: male first name *James John Robert Michael William*
LAST	Person: last name *Smith Johnson Williams Jones Brown*
TTL	Honorific title *President Dr Lord Miss Major*
NORP	Nationality, Religion, Political (adjectival) *American European Indian Republican Christian*
FAC	Facility: names of man-made structures *Broadway Legoland Capitol Boomers SeaWorld*
ORG	Organisation: e.g. companies, governmental *Intel Microsoft Sony IBM Ford*
GPE	Geo-political entity *Canada America China Washington London*
LOC	Locations other than GPEs *Europe Africa Asia Pacific Earth*
DAT	Reference to a date or period *January May Friday Monday Easter*
LANG	Any named language *English Chinese Arabic Spanish Hebrew*

Table 2: The semantic classes

etc. We ignored entity types that were nonsensical without multi-word terms including WORKS OF ART, LAWS and EVENTS. We were also interested in more fine-grained distinctions for the PERSON type, which we split into MALE and FEMALE first names, and LAST names. This resulted in the semantic classes listed in Table 2, which we used for all experiments unless otherwise noted.

We found that the mutual exclusion bootstrapping was most accurate when additional *stop classes* (like stop-lists) were included to help bound the semantic classes. These classes were selected based on observed semantic drift in specific categories. For instance, the JEWEL class was added to stop FEMALE from drifting when it reached names like *Ruby*. The stop classes we included were ADDRESS, BODY PART, CHEMICAL, COLOUR, DRINK, FOOD, JEWELS and WEB terms.

7 Selecting seed lists

To create seed lists we collected named entity lists from a variety of sources. The basis for each collection was the list of most frequent entities for that category from the BBN corpus. This was supplemented with external sources e.g. lists of Fortune 500 companies for ORG; the largest cities from Wikipedia for GPE; and names from the US Census for FEM, MALE and LAST.

We then extracted the frequency of each term in these lists from the Web 1T corpus. Seed lists were created using the top 50, 20, 10 and 5 most frequent single-word terms from these lists. Words that occurred in multiple categories (e.g. *French* in NORP and LANG) were assigned to one category or the other, to ensure each seed list was mutually exclusive. We also created seed lists for the stop classes based on our initial experiments.

8 Evaluation

Our evaluation process involved manually inspecting each extracted term and judging whether it was a member of the semantic class, following Riloff and Jones (1999). To make this more efficient, we stored a cache of previous evaluator decisions for each class so that once a decision had been made for a particular term in a particular class it would be made automatically in future instances.

Although the seed lists were mutually exclusive, for the purposes of evaluation ambiguous words such as *French* were counted as correct if they appeared in either valid category (NORP or LANG). This means that MEB has a minor disadvantage in the evaluation because terms may belong to multiple classes with other approaches.

Evaluation was made more difficult by the fact that we had only single word terms and yet many company names, facility names, etc. are typically multi-word terms. When the single word was an clearly part of a multi-word term we counted it as correct (eg. *Coast* as a LOC). However, if the word was not strongly correlated with the semantic class (e.g. *The* or *Next*) it was not counted as correct. Obvious mis-spellings of words (eg. *Januray*) were also counted as correct. The extracted terms that were unrecognised by the evaluator were checked using Wikipedia and Google.

To compare approaches and parameters we used accuracy at n – the percentage of correct terms in the top n ranked terms for a given category. This evaluation gives a realistic measure of the practical usefulness of the results since the ranked list of bootstrapped terms will be used directly in downstream NLP components. For many experiments this is averaged over the semantic classes (Av(n)). We also we calculated the *inverse rank* (InvR) – the sum of the inverse rank of all correct terms. InvR provides a summary of both the number of correct terms and their ranking in the list.

For comparing the accuracy of different approaches and parameter settings, we manually evaluated all 11 semantic categories down to $n = 50$, which was enough to discriminate between most results. For the final results we evaluated

TYPE	nS	nT	nC	Av(10)	Av(50)
MB	5	5	5	55	21
MB	5	5	10	58	28
MB	5	5	100	79	59
MB	5	5	200	80	**68**
MB	5	5	300	**84**	66
MEB-NS	5	5	5	84	67
MEB-NS	5	5	10	**89**	**68**
MEB	5	5	5	86	67
MEB	**5**	**5**	**10**	**90**	**78**

Table 3: Results comparing approaches.

down to the point where MEB was still producing good results, with a maximum depth of $n = 400$.

9 Results

There are three main parameters to vary in the MEB algorithm – the number of terms in each seed list (nS), and the number of terms (nT) and contexts (nC) to add in each iteration. Our default parameters are 5 for nS, nT and nC. For the experiments below we compare the average semantic class accuracy at 10 and 50 terms.

Table 3 summarises the comparison of mutual bootstrapping (MB) including multi-level bootstrapping, with both mutual exclusion bootstrapping with (MEB) and without (MEB-NS) stop classes. The main results are that MEB significantly outperforms MB and that stop classes play a significant role in bounding semantic classes reducing semantic drift. An interesting new result is that mutual bootstrapping performs badly when few contexts are added, but performs much better when many contexts, e.g. 200, were added in each iteration.

We intend to do further analysis on the many contexts result for MB, but it appears that since MB is very susceptible to semantic drift using many pieces of contextual evidence extracted using the initial seed words is crucial for good performance.

9.1 Parameter settings

For the remainder of the experiments we use MEB with stop classes. In Table 4, we see the results we would expect for increasing the number of seed words for each semantic class. The accuracy is highest when we use 50 seed terms, although collecting 50 seed terms this would take significantly effort than the default of 5. Of course, we can use MEB to extract terms and then manually correct to create larger seed sets quickly.

nS	nT	nC	Av(10)	Av(50)
2	5	5	65	50
5	5	5	86	67
10	5	5	94	67
20	5	5	95	84
50	**5**	**5**	**95**	**91**

Table 4: Results for different seed list size.

nS	nT	nC	Av(10)	Av(50)
5	1	5	86	63
5	2	5	86	69
5	5	5	86	67
5	**10**	**5**	**84**	**70**

Table 5: Results for terms added per iteration.

nS	nT	nC	Av(10)	Av(50)
5	5	1	76	64
5	5	2	77	59
5	5	5	86	67
5	**5**	**10**	**90**	**78**
5	5	15	90	74
5	5	20	90	72
5	5	100	90	62

Table 6: Results for contexts added per iteration.

In Tables 5 and 6 the number of terms or contexts parameters are varied. Adding 10 terms per iteration is more effective than the default of 5, and both outperform the more conservative strategy of only adding one term per iteration. Adding 10 contexts per iteration is also more effective than the one context per iteration used by Riloff and Jones (1999). However, adding 10 terms and 10 contexts per iteration is not as accurate, so the 5–5–10 settings are used for the remaining experiments unless noted.

We investigate the robustness of the results to the quality of the seed sets in Table 7. To experiment with this we created three sets of seed sets with HIGH, MID, and LOW frequency terms as calculated from the Web 1T unigram counts. The HIGH counts are the default set used for the other experiments. We also created a set that was manually selected to best represent the semantic class. This significantly outperformed frequency-based seed sets demonstrating that selecting good seed terms is crucial to high accuracy.

TYPE	nS	nT	nC	Av(10)	Av(50)
HIGH	5	5	5	86	67
MID	5	5	5	90	70
LOW	5	5	5	88	70
MANUAL	**5**	**5**	**5**	**92**	**79**
MANUAL	5	5	10	92	75

Table 7: Results for different seed lists.

9.2 Distributional approaches

Another standard approach to extracting lexical semantic resources is *distributional similarity*, based on the distributional hypothesis that similar terms appear in similar contexts. In distributional approaches, all of the contextual information is summarised in weighted context vectors which are compared using measures of similarity in vector space. We wanted to compare these approaches since this hasn't been done previously using exactly the same data. Hearst and Grefenstette (1992) experimented with combining template methods with the Grefenstette (1994) distributional approach.

We use the distributional similarity approach presented in Curran (2004). The same filtered set of Web 1T 5-grams is converted into context vectors, which corresponds to a window-based context, and the standard t-test weighting and Jaccard measure functions were used (Curran, 2004). Synonym lists of length 200 were generated for head terms that occurred with frequency ≥ 1000.

To map from head terms to semantic classes, we experimented with three methods used in the similar task of supersense tagging (Curran, 2005), where each term from the seed list can vote for synonyms for that class. There are three weighting schemes: with SET each synonym is equally weighted; with SCORE the distributional similarity score weights each synonym; and with RANK the inverse rank weights each synonym. The collected synonyms for each semantic class are then sorted by weighted votes and the top *n* selected.

The results are shown in Table 8. The SET method performs significantly worse than SCORE and RANK, but none of the methods are competitive with the best MEB system on the top 50 extracted terms.

9.3 Semantic Classes

We evaluate the performance of individual semantic classes in Table 9. The evaluation includes accuracy at depths of up to 400 terms per semantic class. We stopped evaluating each semantic class after MEB stopped finding new terms in that class. We have also calculated the inverse rank for the individual classes. The results show that some semantic classes are considerably more difficult than others, showing drift after far fewer iterations than other classes. This evaluation is harsh on classes with fewer than 400 terms, e.g. honorific titles. For the four most reliable classes we also manually checked down to 750 terms, where MEB still performed extremely well with FEM 63%, MALE 88%, LAST 95% and GPE 96%.

TYPE	Av(10)	Av(50)
SET	67	58
SCORE	86	70
RANK	**88**	**72**

Table 8: Results for distributional similarity.

Some pairs of semantic classes, especially FAC and ORG, and LOC and GPE, require much more subtle semantic distinctions than previous bootstrapping evaluations. The evaluators had considerable difficulty distinguishing between a facility and an organisation based on single-word terms. We merged these problematic categories into more general categories to see if this improved the results. We merged FAC and ORG to form the FOG class, and LOC and GPE to form PLACE.

The two merged classes appear in Table 9. Merging improved the performance dramatically with FOG and PLACE 95% and 100% accurate (respectively) at 400 extracted terms. However, we noticed a slight decrease in performance for the NORP and DATE which demonstrates the boundary interactions that can occur with MEB.

9.4 Resource coverage

There is a suspicion that automatically extracted lexical semantic resources tend to contain the same terms that are available in existing manually created resources. By using existing resources to speed up the manual evaluation process we were able to identify interesting terms that would typically not be contained in existing resources, e.g.:

- foreign translation terms. MEB found non-English months including *Oktober* and *Chwefror* (February in Welsh);
- names missing from the US census lists, which covered names down to 0.001% of the population, e.g. *Uday* and *Igor*;
- programming languages, e.g. *Python*;

n	FEM	MALE	LAST	TTL	NORP	FAC	ORG	GPE	LOC	DAT	LANG	FOG	PLACE
10	100	100	100	100	90	70	50	100	80	100	100	100	100
20	100	100	100	100	90	60	35	100	80	100	100	100	100
50	100	100	100	66	90	48	16	100	64	78	100	100	100
100	99	100	100	51	67	32	8	100	39	56	85	100	100
150	99	100	100	38	61	23	5	99	31	65	66	100	100
200	95	100	100	31	57	18	-	99	27	60	63	99	100
250	91	100	100	27	49	-	-	98	22	-	58	99	100
300	91	97	100	-	42	-	-	98	-	-	58	97	100
350	88	94	100	-	-	-	-	98	-	-	53	95	100
400	87	94	99	-	-	-	-	98	-	-	47	95	100
InvR	5.92	6.27	6.30	4.35	4.95	3.09	2.39	6.26	3.89	4.58	5.38	6.50	6.57

Table 9: Results for our 11 original categories. The maximum inverse rank possible is 6.57.

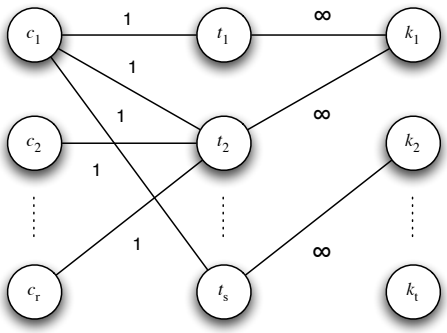


Figure 1: Multi-partite graph for MEB.

- many rare languages - Aboriginal and African tribal languages, and *Klingon*!

10 MEB as Multi-way cut

Mutual exclusion bootstrapping can be posed as a multi-partite graph partitioning problem where semantic classes, terms and contexts are nodes and membership and cooccurrence for the edges. Theoretically, this approach allows terms and contexts to be optimally separated into semantic classes.

Figure 1 shows the multi-partite graph. Given the set of contexts C and the set of words T, the *word-context relation* $R \subseteq C \times T$ denotes pairs (c,t) for which term t appears in context c.

A word/context u is *connected* to word/context v, if there exists a path from u to v in graph $G\langle T \cup C, R\rangle$. A *seed semantic class* $\Gamma : K \rightarrow 2^T$ is a partial mapping from semantic class to a subset of terms. A word/context $u \in T \cup C$ is *associated* with semantic class k, if there exists term $t \in \Gamma(k)$ that is connected to u.

We seek for a *word/context labelling* $\Lambda : T \cup C \rightarrow K$ such that a minimal number of pairs are to be removed from R to make the classification unique, i.e. there exists neither a context nor a term for which we have multiple semantic class associations. Intuitively this corresponds to splitting the terms and contexts into mutually exclusive semantic classes by ignoring the minimum number of occurrences of terms with contexts.

MEB is reducible to a multi-way cut. For the reduction we construct a multi-partite graph as shown in Fig. 1. The first and second node layers represent the term-context relationship and the second and third node layers represent the seed semantic class mapping. The semantic classes are the multi-way cut terminal vertices and the multi-way cut of the multi-partite graph is optimal MEB.

10.1 Multi-Way Cut

Given a graph $G\langle U,E\rangle$ and a set $T \subseteq V$ of k terminal vertices, a *multi-way cut* (also known as k *way-cut*) (cf. (Bachour et al., 2005)) is a set $C \subseteq E$ of edges such that in $G'(V, E-C)$, no path exists between any two nodes of T, i.e., the terminal vertices become disconnected from each other. The multi-way cut problem seeks for a cut such that $|C|$ becomes minimal. The weighted multi-way cut problem seeks a cut C such that $\sum_{e \in C} w(e)$ is minimal where $w(e)$ is the weight of edge e.

For $k = 2$, the problem is reduced to the $s-t$ min-cut problem introduced by Ford and Fulkerson (Calinescu et al., 1998) that can be solved via its dual problem – the *max-flow* problem in polynomial time. Unfortunately, for undirected graphs the multi-way cut problem is NP-hard for $k \geq 3$. Dahlhaus et al. (1994) give a simple combinatorial isolation heuristic that approximates a solution with error bounded by $2 - \frac{2}{k}$ to the optimal solu-

tion. In this algorithm $k-1$ terminals are chosen and a $s-t$ min-cut separates the selected terminal from the other terminals. The union of these cuts give the approximation of the multi-cut.

The approximation algorithm in Dahlhaus et al. (1994) has the worst approximation bound but the the best worst-case complexity class. A deterministic algorithm for max-flow (Goldberg and Tarjan, 1988) results in a worst-case complexity of $\tilde{\mathcal{O}}(k \cdot m \cdot n)$ where n is the number of vertices and m the number of edges in graph G. A probabilistic algorithm for $s-t$ min-cut even improves the worst-case complexity to $\tilde{\mathcal{O}}(k \cdot m)$.

We have completed a practical implementation of $s-t$ min-cut MEB that can run on datasets of around 10 000 terms and the results are very promising. We believe that posing MEB as an optimal graph partitioning problem has great potential to improve the quality of our results further.

11 Conclusion

The MEB algorithm deserves further study as do the many contexts results for the existing mutual bootstrapping algorithm. For instance, the results may be sensitive to the ordering of semantic classes, and to the ranking of terms and contexts. Also, the results are dependent on the ambiguity and representativeness of the initial seed list for both semantic classes and stop lists. Since evaluation is very time consuming we haven't explored these problems yet. We would also like to investigate whether the mutual exclusion can be relaxed to some degree without losing the significant gains in performance. Finally, we hope to apply MEB to other tasks (e.g. common nouns) and languages.

In this paper we have proposed *mutual exclusion bootstrapping* (MEB), based on the mutual bootstrapping algorithm proposed by Riloff and Jones (1999), which attempts to overcome the semantic drift common to iterative bootstrapping techniques. MEB extracts terms and contexts for multiple semantic classes in parallel, imposing a strict constraint that the classes must be mutually exclusive with respect to both terms and contexts.

Although this assumption is false for many pairs of semantic classes, it still significantly improves the quality of the extracted terms. We have evaluated our approach on a wide range of proper-noun named-entity classes using the massive Google Web 1T dataset, also demonstrating that MEB scales efficiently. We have also experimented with a wide range of parameters that affect bootstrapping accuracy. The result is an algorithm that can extract large lexical semantic resources with a high degree of reliability. Finally, we have demonstrated that MEB can be posed as the multi-way cut optimisation problem from graph theory, solvable using approximation algorithms.

Acknowledgements

We would like to thank the anonymous reviewers and members of the LTRG at the University of Sydney, for their feedback. James Curran and Tara Murphy were funded on this work under ARC Discovery grants DP0453131 and DP0665973.

References

Eugene Agichtein, Eleazar Eskin, and Luis Gravano. 2000. Combining strategies for extracting relations from text collections. Technical Report CUCS-006-00, Department of Computer Science, Columbia University, New York.

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM Conference on Digital Libraries*, pages 85–94. San Antonio, TX USA.

Khaled Bachour, Eda Baykan, Wojciech Galuba, and Ali Salehi. 2005. Citation network partitioning. Technical report, Ecole Polytechnique Fédérale de Lausanne.

Matthew Berland and Eugene Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 57–64. College Park, MD USA.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Technical Report LDC2006T13, Linguistic Data Consortium.

Gruia Calinescu, Howard Karloff, and Yuval Rabani. 1998. An improved approximation algorithm for multiway cut. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 48–52. ACM Press, New York, NY, USA.

Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 120–126. College Park, MD USA.

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh, Edinburgh, UK.

James R. Curran. 2005. Supersense tagging of unknown nouns using semantic similarity. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 26–33. Ann Arbor, MI USA.

Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, P. D. Seymour, and Mihalis Yannakakis. 1994. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894.

A.V. Goldberg and R.E. Tarjan. 1988. A new approach to the maximum flow problem. *J. of the ACM*, 35(4):921–940.

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

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th international conference on Computational Linguistics*, pages 539–545. Nantes, France.

Marti A. Hearst and Gregory Grefenstette. 1992. A method for refining automatically-discovered lexical relations: Combining weak techniques for stronger results. In *Statistically-Based Natural Language Programming Techniques: Papers from the AAAI Workshop*, Technical Report WS-92-01, pages 72–80. AAAI Press, Menlo Park.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 474–479. Orlando, FL USA.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124. Providence.

Brian Roark and Eugene Charniak. 1998. Noun-phrase co-occurrence statistic for semi-automatic semantic lexicon construction. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th annual meeting of the Association for Computational Linguistics*, pages 1110–1116. Montréal, Québec, Canada.

Neel Sundaresan and Jeonghee Yi. 2000. Mining the web for relations. In *Proceedings of the 9th International World Wide Web Conference*. Amsterdam, Netherlands.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical Report LDC2005T33, Linguistic Data Consortium.

Analogical Mapping Method and Semantic Categorization of Japanese Compound and Complex Sentence Patterns

Satoru Ikehara
Tottori University,
Tottori-city, 680-8552 Japan.
ikehara@ike.tottori-u.ac.jp

Masato Tokuhisa
Tottori University,
Tottori-city, 680-8552 Japan.
tokuhisa@ike.tottori-u.ac.jp

Jin'ichi Murakami
Tottori University,
Tottori-city, 680-8552 Japan.
murakami} @ike.tottori-u.ac.jp

Abstract

To overcome the limit of the conventional machine translation (MT) method based on *compositional semantics*, we proposed an *Analogical Mapping (AM) method* based on *Semantic Typology* and built a semantic category system for Japanese compound and complex sentences. The *AM-method* maps linguistic expressions into other expressions with the same meaning with semantic categorization (based on concepts called *Truth Items*). We also built a semantic category system composed of two sub-systems: one for classifying the meanings (222 categories) represented by the relation between two clauses and the other for classifying the meanings (740 categories) represented by clauses. These semantic codes were assigned to sentence patterns (SPs) (226,800 patterns) registered in the sentence pattern (SP) -dictionary we recently developed. We ascertained that these were useful for selecting semantically correct candidates from matched patterns in input sentences.

1. Introduction

Significant investment has been made in machine translation (MT), resulting in noteworthy achievements (Nakamura, 1983; Nagao et al., 1998; Tanaka, 1998). However, it is very difficult to develop high quality MT systems between languages belonging to very different families, such as Japanese and English.

Most practical MT systems so far have been based essentially on the *transfer method*, which is, in turn, based on *compositional semantics.* A problem with this method is that it produces translations by separating the syntactic structure from the meaning and is thus liable to lose the meaning of the source text.

Much attention has been focused on the use of *cognitive grammar* (Langacker, 1991: Lakoff, 1986), and *construction grammar* (Fillmore et al. 2003) in hopes of solving this problem. In these methods, various measures are taken to associate the meaning of constituents with the overall meaning of a sentence. However, the standards for determining the structural meaning units are undefined.

Better quality in translation can be achieved from *pattern-based MT* (Takeda, 1996) where the syntactic structure and semantics are handled together. However, this method requires immense sentence pattern (SP) dictionaries, which are difficult to develop, and so far, this method has only been used in hybrid systems where small-scale SP-dictionaries for specific fields are used to supplement a conventional *transfer method.*

Example-based MT (Nagao, 1984; Sato, 1992; Brown, 1999) might solve this problem. This method obtains translations by substituting semantically similar elements in structurally matching translation examples; hence, there is no need to prepare an SP-dictionary. However, the substitutable constituents depend on translation examples. This makes it impossible to determine them in real time. This problem could be addressed by manually tagging each example beforehand, but the resulting method would be just another *pattern-based MT.*

A *Multi-Level-Translation Method* (Ikehara et al., 1987) has been proposed and an SP-diction-

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 181–190,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

ary called *A-Japanese-Lexicon* has been developed for simple Japanese sentences (Ikehara et al., 1997) to address this problem, This dictionary includes 17,000 valence patterns of relation between verbs and case elements. Semantic use of nouns (400,000 words) is specified using *semantic attributes* (2,700 types). This dictionary has significantly improved the quality of translation of simple Japanese sentences into English.

Subsequently, a new language model that focuses on non-compositional linguistic expressions has been proposed, and a large-scale SP-dictionary (226,800 SP pairs) has recently been developed for Japanese compound and complex sentences (Ikehara, 2001).

The SPs registered in this dictionary are considered meshes for filtering out the meanings of Japanese expressions. Then, we developed an *Analogical Mapping method* (*AM-method*) that maps linguistic expressions into other expressions with the same meaning via concepts (called *Truth Items*) and built a semantic category system for the meanings of Japanese compound and complex sentences. The *AM-method* is an ideal MT method that uses an SP-dictionary.

2. *Analogical Mapping Method*

The relation between syntax and semantics has been one of the most controversial issues in language translation. In this section, we describe the *AM-method,* which is based on *Semantic Typology* (Arita 1987) and *Analogical Mapping theories* (Ichikawa, 1963) .

2.1 *Semantic Typology* of Expressions

Natural language has many “forms" of expressions, so speakers can express subtly different concepts. These forms are also used as a framework in the process of conceptualizing objects.

Jun Arita (1987) a Japanese linguist specializing in German, proposed the idea of *Semantic Typology* and *typological semantic units*. He says that *typological semantic units* exist one level below specific linguistic expressions. They are semantic structures that have been abstracted or simplified to the maximum extent practical without changing their meaning, and they are also considered as a mesh for filtering out concepts from linguistic expressions. Linguistic expressions can be analyzed and comprehended using these semantic units.

Based on this idea, translation can be considered as a process of "filtering out” cognition presented by speakers using semantic units of a source language and reproducing them in other semantic units of a target language. To produce high-quality translations, a translator must have meshes of filtering out concepts for the source language that exactly corresponds to those of the target language. If the exact grid is not available, that is, if a concept that exactly corresponds to the concept appearing in the source language is not found in the target language, it may be represented by combining more concrete or similar concepts. A large-scale SP-dictionary has recently been developed (Ikehara et al., 2006) for such a typological semantic unit.

2.2 *Analogical Mapping Theory*

Kikuya Ichikawa (1963) formulated the analogical reasoning in scientific discovery and then developed his *Analogical Mapping Theory* [*1] in *Creative Thinking*, referred to as the *Theory of Equivalent Transformation,* which states that analogical thinking lies at the core of human creativity. This theory presented a sort of model for the process of solving creative problems by assuming that different systems may have a commonality, ε, in their events or phenomena under a certain condition, C. Equation (1) shows this assumption.

$$C\ (\ A_{\alpha} \overset{\varepsilon}{=} B_{\beta}\ ). \qquad (1)$$

where C is a condition, ε is a commonality,
A_{α} is an event in System α, and
B_{β} is an event in System β.

*1 In the field of artificial intelligence, human intelligence was traditionally thought to be essentially comprised of human's generic thinking rules and the reasoning (inference) ability supported by these rules. However, more attention has been paid to roles of human's analogical ability since the 1980's in AI (Suzuki, 1996).

Analogical thinking refers to the process where, given an event, A_α (source), in System α, humans conjure up an event, B_β (target), in System β that has a commonality, ε, under a condition, C.

Translation is a process that involves understanding the meaning of linguistic representations in the source language, remembering their equivalents in the target language and then selecting an appropriate expression. Thus, translation may also be based on analogical thinking, which uses eqation (1) as its assumption.

Equation (1) can be applied to any language, resulting in the following explanation. For us to translate an expression, A_α, in language α into an expression, B_β, in language β, language β must have the expression, B_β that implies a concept represented by the expression, A_α. This logic provides a basis for implementing translation between different languages based on meanings. That is, if the commonality, ε, is considered as a concept that exists in both the source and target languages, translation by semantic units is feasible.

2.3 *AM-method*

(1) Principle of *AM-method*

It is technically difficult to map the countless individual linguistic expressions of a language onto those of another language with their meanings correctly translated. However, the infinite number of expressions can be reduced to a finite number of semantic units. This method is called the *AM-method*[*1], and uses semantic units as previously discussed. Relation (2) represents the fundamental principles of the method,

$$A_\alpha \Rightarrow C(A_\alpha) \Rightarrow \varepsilon \Rightarrow C(B_\beta) \Rightarrow B_\beta, \qquad (2)$$

Where $\Rightarrow$ is a projection or mapping, ε is a *Truth Item* (a member of a *logical semantic category)* and C is a function to typify a linguistic expression as an appropriate basic semantic unit.

Relation (2) is applied to a translation if $\alpha \neq \beta$ and to rewording in the same language if $\alpha = \beta$. Although this relation represents the translation process of typological semantic patterns, i.e., the non-compositional constituents between the two languages, it can be used for processing compositional constituents.

(2) Logical Semantic Category system

As shown in Fig. 1, the semantic units of the two languages are mapped via a *Logical Semantic Category system (LSC-system).* This system is a set of concepts called *Truth Items.*

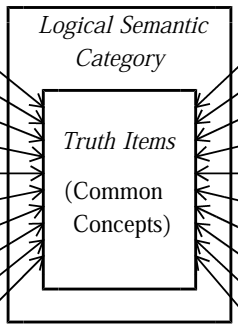
(3) Procedure of *AM-method:*

The AM-method consists of the following steps.

Step 1: Retrieval of matched SPs.

Retrieve the SPs matched to an input sentence from the SP-dictionary.

#	Japanese SPs= C (A α)
1	X1 wa, X2 ga X3 suruyou X4suru
2	X1 wa, X2 ga, taihenn X3 nanode X4 dekinai
3	X1 wa X2 ga X 3suruto ikenainode, X4 suru
4	X1 wa X2 suruto ikenainode X 3sita
5	X1 wa X2 sinaiyou X3 sita
6	mosi X1 ga X2 sitara, X3 wa X4 suru
7	X1 ga X2 sitara X3 wa X4 sita
8	X1 ga X2 sitatoki X3 wa X4 sita
9	X1 ga X2 surunara X3 wa X4 sitemoyoi
0	X1 wa X2 nanode X3 da



#	English SPs= C (B α)
1	X1 X4 so that X2 X3
2	X2 is so X3 that X1 cannot X4
3	X1 X4 for fear that X2 X3
4	X1 X3 not to X2
5	X1 is X3 for X1 is X2
6	X3 X4 in the case X1 X2
7	When X1 X2, X3 X4
8	If X1 X2, X3 X4
9	If X1 X2, X3 may X4
0	X3 may X4 provided that X1 X2

Fig. 1 Semantically equivalent mapping via *Truth Items*

*1 The translation procedure of this method is not deterministic. It is essentially different from *Example-base* or *Analogy-base* MT as well as conventional *Transfer Method.* We call it *Mapping Method* in stead of *Transfer Method.*

Step 2: SP mapping with *Truth Items.*

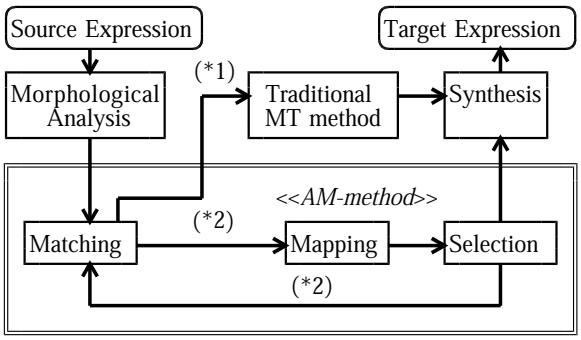
Matched SPs are mapped into semantically equivalent SPs by using *Truth Items.*

Step 3: Expression Generation by SP.

Target expressions are generated from the mapped SPs.

(4) Application to MT

The AM-method is used mainly with *non-compositional expressions*[*1] that cannot be translated by a method based on *Semantic Composition*. Because *compositional expressions* can be translated using conventional methods, both of these methods can be used together, as shown in Fig. 2. The closed loop indicates the *AM-method* is recursively used for different levels of included expressions, such as *non-compositional* clauses and phrases.



(*1) Compositional expressions
(*2) Non-compositional expressions

Fig. 2 MT System incorporated *AM-method*

3. *Logical Semantic Category system*

3.1 Target SP-dictionary

A large-scale SP-dictionary was recently developed for Japanese compound and complex sentences with two or three clauses based on the *Non-compositional language model* (Ikehara et al., 2004: 2006). Three kinds of SPs were generated from a Japanese to English parallel corpus through generalizing *compositional constituents,* as shown in the following three steps.

(a) *Word-level* SPs: Compositional independent words (nouns, verbs, etc.) were converted into *word variables.*
(b) *Phrase-level* SPs: Compositional phrases (noun phrases, verb phrases, etc.) were converted into *phrase variables.*
(c) *Clause-level* SPs: Compositional clauses were converted into *clause variables.*

The number of SPs is shown in Table 1. And their coverages are shown in Table 2. Examples of SPs will be shown later (see Fig 5)

Table 2 Coverage of SP-dictionary

Type of SP	Syntactic Coverage	Semantic Coverage
Word Level	72.1 %	55.0 %
Phrase Level	87.0 %	70.0 %
Clause Level	98.0 %	71.0 %
Total	98.5 %	79.5 %

Table 1. Number of SPs

Sentence Type	SP type	No. of Clauses	Subordinate Clause	Type of SP			
				word-level	*phrase-level*	*clause-level*	Total
Compound	Type 1	2	1 continuous clause.	53,508	36,002	17,859	109,369
	Type 2	3	2 continuous clauses	5,663	3,241	314	9,218
Complex	Type 3	2	1 adnominal clause	42,485	28,040	4,998	75,523
	Type 4	3	2 adnominal clauses	5,638	4,009	780	10,427
Mixed	Type 5	3	both types of clauses	12,510	8,146	1,524	22,280
	——	—	Total	121,904	79,438	25,475	226,817

*1 According to (Ikehara et al., 2006), *Compositional constituent* is defined as a constituent which is interchangeable with other constituents without changing *the meaning of an expression structure.* All other constituents are *Non-compositional constituents.* Based on this idea, *Compositional expression* is defined as an expression consisting of *Compositional constituents,* and *Non-compositional expression* is defined as an expression comprising one or more *Non-compositional constituents.*

We developed a *Logical Semantic Category* system to semantically classify these SPs.

3.2 Design Condition of *Truth Items*

(1) Commonality of *Truth Items*

Logical Semantic Category System consists of a set of *Truth Items. Truth Items* are defined as concepts common to the source and target languages. In general, however, perception is different from language to language even if the same object is concerned. Concepts represented by an expression are not always common to all languages.

This problem has been resolved in the SP-dictionary. Because the SPs of the source and target languages registered in the parallel corpus have a one-to-one correspondence, the concepts represented by the SP of the source the language are approximately the same as those of the target language. For this reason, assuming that the concepts represented by Japanese expressions are common to English expressions, the concepts represented by Japanese expressions are classified and used as the *Truth Items.*

(2) Granularity of Meanings

Consider to design the *Logical Semantic Category System* for Japanese compound and complex sentences, because the granularity of *Truth Items* needs to be fine enough to classify the meanings of the large-number of SPs, as shown in Table 1, it is difficult to define the meaning of an SP by using only one *Truth Item.* We defined the meaning of an SP by using multiple *Truth Items.*

Our design conditions for the system of *Truth Items* are as follows:

a) The meanings of the dependent relation between two clauses are classified by *Semantic category for clause to clause (C-to-C) relation.*
b) The meanings of clauses that compose compound and complex sentences are classified by *Semantic category for clause (C)*
c) *Truth Items* in these systems are hierarchically organized.

3.3 Construction of *LSC-system*

(1) *Semantic category for C-to-C relation*

Masuoka and Takubo (1992) classified dependent clauses into 4 types: *noun, adnominal, continuous,* and *parallel.* They analyzed the meanings of these dependent clauses and subdivided them into about 30 categories.

Table 3. *Semantic category for C-to-C relation*

L1	L2	L3	L4	Total
Compliment Clauses	noun clause	4	10	15
	interrogative clause	2	0	3
	quoted clause	2	7	10
	others	0	0	1
Noun Clauses	complemental	2	0	3
	substantial	0	2	3
	abridged modify	0	7	8
	functional	4	0	5
	"of" type	0	3	4
	others	0	0	1
Adverbial Clauses (see Fig.3)	time	2	16	19
	causality	**3**	**12**	16
	condition/concession	5	11	17
	circumstances	2	8	11
	reverse conjunction	0	9	10
	target	0	8	9
	extent	0	12	13
	premise	0	2	3
	method	0	5	6
	relation	0	7	8
	correlation	0	2	4
	descision	0	6	7
	scene	0	6	7
	authorization	0	3	4
	independent	0	6	7
	others	5	5	11
Parallel Clauses	normal parallel	6	4	11
	revers parallel	0	0	1
	others	0	0	1
others	0	0	0	1
5	29	37	151	222

(c.f.): L1-L4 represent the levels of categories. The number of each cell represents the number of categories at that level.

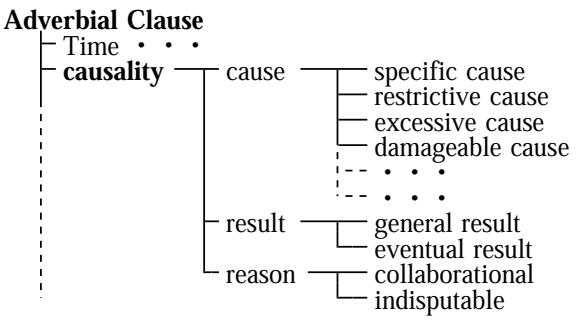


Fig. 3 Example *of Semantic Categories for C-to-C*

Referring to their study, we analyzed the meanings of the relations between clauses contained in thousands of example sentences from the view points of *temporal, spatial, logical,* and *psychological relations*, and constructed a *Semantic category for C-to-C relation*. Our *Semantic category for C-to-C* relation consists of four ranks and 222 types as shown in Table 3. Here, eamples of lower level categories under *causality* of *adverbial clause* are shown in Fig.3.

(2) *Semantic category for C*

A clause corresponds to a simple sentence, which expresses an individual event. The expression of a simple sentence is generally articulated into a propositional part and others, such as tense, aspect, and modality. The meanings of parts other than propositional part are represented by the SP structure. Subsequently, classified simple sentences by the meanings of their propositional parts and designed a *Semantic category for C.*

Simple Japanese sentences can be classified into *verb, adjective,* and *noun sentences* (Teramura, 1982) based on the types of predicates, We constructed three semantic attribute systems and used them to classify simple sentences.

Table 4. *Semantic category for C*

L1	L2	L3	L4	L5	Total
Verb Clause (see Fig.4)	**perception/emotion**	3	15	26	45
	intellectual act	6	17	20	44
	act of daily life	4	17	10	32
	act of society life	4	8	0	13
	social activity	7	23	9	40
	phenomena	9	29	11	50
	change	5	7	6	19
	movement	7	16	0	24
	objective action	4	15	8	28
Adjective clauses	nature prescription	6	34	34	75
Noun Clauses	subjective	4	26	16	47
	place	6	23	0	30
	concrete object	6	40	0	47
	abstract object	12	27	0	40
	event	4	38	98	141
	abstract relation	9	55	0	65
3	16	96	390	238	740

(c.f.:) L1-L5 represent the levels of categories. The number of each cell represents the number of categories at that level.

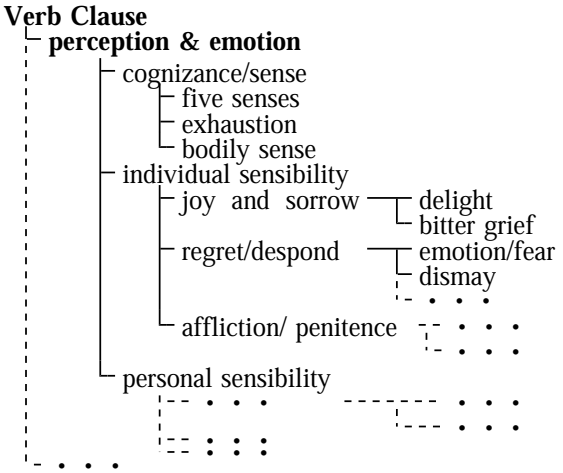


Fig. 4 Classification of perceptional or emotional Expression

The *Semantic category for C* is shown in Table 4. Here, examples of lower level categories under *perception/emotion* of *Verb Clause* are shown in Fig. 4.

Based on this classification system, we also developed a semantic verb dictionary for 6,000 words and a semantic noun dictionary for 60,000 words.

3.4 SP classification by Semantic Code

(1) SP types and semantic code

As previously stated, the SP-dictionary consists of three levels of SPs. Out of these, the *phrase level* and *clause level* SPs were obtained by further generalizing the *word level* SPs. Therefore, *Truth Items* of *word level* SPs are precursor to *phrase level* and *clause level* SPs.

Table 5. SP type and Semantic code

Semantic code	*Semantic category for C-to-C*		*Semantic category for C*		
Type of SPs	1st sub-ordinate clause	2nd sub-ordinate clause	1st sub-ordinate clause	2nd sub-ordinate clause	main clause
Type 1	○	—	○	—	○
Type 2	○	○	○	○	
Type 3	○	—	○	—	
Type 4	○	○	○	○	
Type 5	○	○	○	○	

(c.f).: Circle means applicable items

Table 5 shows the relations between the types of SPs and semantic codes. Columns 1 and 2 represent semantic codes defined by the *Se-*

mantic category for C-to-C, and columns 3 to 5 represent those of the *Semantic category for C.*

(2) Semi-automatic classification

A very large number of SPs is registered in the SP-dictionary. The meanings represented by the relation between clauses are not determined solely from conventional analyses, such as morphological and syntactic analyses. A great deal of human labor is required to assign semantic codes from the *Semantic category for C-to-C relation* to them.

This work was semi-automated by noticing that some corresponding relations can be observed between the meanings and the forms of expressions. In the case of compound sentences, we noticed the roles of conjunctive particles and functional words, and the relations between syntactic attributes of main clauses and subordinate clauses. For complex sentences, on the other hand, we noticed the relations between the type of embedded clause (inside/outside relation) and the meaning of antecedents when the complex sentence contained them.

Based on this information, we developed 90 templates for classifying the meanings of SPs. With these templates, all SPs were classified into approximately 90 groups. Semantic codes were then manually assigned to every SP in the group.

In contrast, it is not difficult to assign semantic codes of the *Semantic category for C,* because these are determined by the meanings of predicate parts of clauses. This work was automatically done by using the results of morphological and syntactic analyses and the semantic word dictionary as mentioned in 3.2 (2).

Explanation		Contents registered in SP-dictionary
Example sentence	Japanese	*kokono kikouwa watasini atteirunode rougowa kokode kurasitai* ここの 気候は わたしに 合っているので 老後は ここで 暮らしたい。
	English	The climate here suits me, so I would like to live here in old age.
Word Level SP	Japanese	/y$1^{/tcfkN1(6810)の/kN2(15421)は}/tcfkN3(11110, 11112, 11120, 11160, 11211)に$1/cfV4(5110, 6930, 6940).teiru^rentaiので</ycN5は>!老後は/cfADV6(9900)/fV7(3210, 3240, 6970).tai。
	English	N2 ADV(N1)V4 N3^obj, so <I|N5> would like to V7^base ADV6 in old age.
Phrase Level SP	Japanese	/ytcfkNP1(15420, 15421)は!VP2(5110, 6930, 6940).teiru#6(.genzai|.kako)^rentaiので</ycN3は>!VP4(3210, 3240, 6970).tai。
	English	NP1 VP2#6(^present|^past), so <I|N3> would like to VP4^base.
Clause Level SP	Japanese	/ytcfkCL1(5110, 6930).teiru#5(.genzai|.kako)^rentaiので!CL2(3210, 6970)。
	English	CL1, so CL2^past.
Semantic code		FUb100 ／－ ‖ 4110, 5930, 5940-2210 ／－／ 2240, 5970 ‖

[Explanatory notes]

Word-level SPs: ① N1, N2, N3, N5: *Noun variables.* ② V4, V7: *Verb variables.* ③ ADV6：*Adverb variable*, Here, attached bracket represents semantic attribute numbers specifying semantic constraints on a variable. ④$1^{}: constituents which canbe moved, ⑤.teiru, .tai :*tense/aspect/modaliy function*, ⑥ ^rentai, ^base, N3^ob: *word form function*, ⑦ ADV(N1) *part of speech change function*, ⑦<I|N5>: *constituent selection symbol*, ⑧ /ytcfk: Place of a constituent that may appear. (y = adonominal clause, t = constinent clause, c = case element, f = adverbial constituent, k = adonominal constituent)

Phrase-level SPs: ① NP1: *Noun phrase variable.* ② VP2, VP4: *Verb Phrase variable,* ③ VP2 #6 (^present| ^past): *selective tense description*

Clause-level SPs: ① CL1, CL2: *Clause variable*

Semantic Code: ① FUb100：*Semantic category for C-to-C* (causality-cause), ② 4110, 5930, 5940-2210: *Semantic category for C* (Subordinate clause), ③ 2240: *Semantic category for C* (Main clause)

Fig. 5 Example of semantic code assigned to SP

Fig. 5 shows an example of an SP (type 1) in which semantic codes were assigned as in Table 5..

4. Application to Selection of Matched SPs

Truth Items can be used to select SPs from matched SPs for an input sentence. When we retrieve syntactically matched SPs from the SP-dictionary, many SPs are usually obtained for one input sentence. These SP candidates include many semantically incorrect SPs. *Truth Items* are expected to be used for selecting semantically correct candidates. This section shows an example.

(1) Experimental method

We prepared two kinds of SP search programs as follows:

(a) Pattern searcher
This program compares the constituents of an input sentence and those of the patterns registered in the SP-dictionary and extracts syntactically matched SPs.

(b) Semantic searcher
This program retrieves semantically matched SPs from the SP-dictionary by comparing *Truth Items* of SPs and those of an input sentence.

The *pattern searcher* guarantees corresponding relation of constituents between an input sentence and a matched pattern, so the target expression can be generated from matched patterns. These relationships, on the other hand, are not guaranteed with the *semantic searcher*, so target expressions are not always generated from retrieved patterns. In this experiment, the SP candidates obtained by the *pattern searcher* were narrowed down with the *semantic searcher.*

(2) Experimental results

The number of SPs retrieved with each searcher for the following input sentence is shown in Table 6.

Input sentence:
watashiwa yuujinwo tatotte joukyousita
私は　友人を　頼って　上京した。
(I came to Tokyo, looking to my friend for assistance.)

Table 6. Number of retrieved SPs

Patten search program	*Word-lv.*	*Phrase lv.*	*Clause lv.*	Total
Pattern searcher	363	2,774	395	3,532
Semantic searcher	15	8	0	23
No. of common SPs	6	8	0	14

As shown in this table, many SP candidates (3,532 in total) were obtained with the *Pattern searcher*. Out of these, the top five candidates are shown below in decreasing order of the number of matched literal constituents (the order seemed to be correct).

SPs retrieved with *Pattern searcher*

△(1) /y$1/tcfkN1 を/cfV2(て|で)$1^{/ytckN3 は}/cf(V4.kako|ND4 をした)。→ N3 V(V4|ND4).past when N3 V2.past N1.
<例>話を聞いて彼は逆上した*(hanasiwo kiite karewa gyakujousita)*。"He went wild when he heard that."

×(2) /y$1/tcfkN1 を/cfV2(て|で)$1^{/ytckN3 は}/cf(V4.kako|ND4 をした)。→ N3 be.past V(V4|ND4).past to V2 N1.
<例>それを聞いて私は安心した*(sorewo kiite wa tasiwa ansinsita)*。"I was relieved to hear it."

×(3) /y$1/tcfkN1 を/cfV2(て|で)$1^{/ytckN3 は}/cf(V4.kako|ND4 をした)。→ V2^grn N1 V(V4|ND4).past.
<例>死体を見て彼は茫然自失した*(sitaiwomite karewabouzenjisitsusita)*。"Seeing the dead body freaked him out."

×(4) /y$1/tcfkN1 を/cfV2(て|で)$1^{/ytckN3 は}/cfV4.kako 。→ With N(V2) of N3^poss N1 N3 V4.past.
<例>手を振って彼女は立ち去った*(tewofutte kanojowa tachisatta)*。"With a wave of her hand she went away."

×(5) /y$1/tcfkNP1 を/cfV2(て|で)$1^{/ytckN3 は}!VP4.kako 。→ N3 V2.past NP1 to VP4.
<例>盲目という悪条件を克服して彼は偉大な学者になった(mekuratoiu akujoukennwo kokufukusite karewa idaina gakushani natta)。"He overcame the handicap of blindness to become a great scholar."

The meanings of the marks (○、△、×) are as follows:

○：easy to generate a good translation
△：semantically correct but not easy to generate a good translation
×：semantically incorrect and unusable for translation

As shown in these examples, many SPs were retrieved with the *Pattern searcher*, but very few are semantically correct.

Twenty-three SP candidates were retrieved with the *Semantic searcher*. Out of these, the top five candidates are shown below in the same order as mentioned above.

SPs retrieved with *Semantic searcher*

△(1) /y</tkN1 は>/cf いささかの/k 知辺を/cf 頼って/ytckN2 へ/cf(上っ|のぼっ|上ぼっ)た。→ <I|N1> went to N2, looking to a slight acquaintance for assistance."
<例>いささかの知辺を頼って都へ上った*(isakakano chiennwo tayotte miyakoni nobotta)*。"I went to town, looking to a slight acquaintance for assistance."

○(2) /y$1^{/tcfkN1 は}/tcfkN2 を/cf 頼って$1/ycf 上京した。→ N1 came to Tokyo from the country counting on N1^poss N2's help.
<例>わたしはおばを頼って上京した*(watasiha obawo tayotte joukyousita)*。"I came to Tokyo from the country counting on my aunt's help."

○(3) /y</tkN1 は>/tcfkN2/tck 一人の/k 知人を/c 頼って/ycf 上京した。→ <I|N1> came to town, looking to an AJ(N2) friend for assistance.
<例>たった一人の知人を頼って上京した*(tattahitorino chijinnwo tayotte joukyousita)*。"I came to town, looking to an only friend for assistance."

○(4) /y</tkN1 は>/tcfkN2 を/cf 頼って</ycfN3 は>/cf (V4.kako|ND4 をした)。→ <I|N3>N(V4|ND4) where <I|N1> could rely on <my|N1^pron^poss> N2.
<例>親類を頼って上京した*(sinnruiwo tayotte joukyou sita)*。"I went to Tokyo where I could rely on my relatives."

×(5) /y</tkN1 は>/tcfk 雲の/k 中を/tcfkN2 に/cf 頼って/ycfV3.kako。→ <We|N1>V3.past through the clouds with the help of N2.
<例>雲の中を計器に頼って飛んだ*(kumononaka wo keikini tayotte tonda)*。"We flew through the clouds with the help of the instruments."

From the results of these two experiments, the SP candidates can be narrowed down to 16. Out of these, the top five candidates are shown below.

Common SPs

○(1) /y$1^{/tcfkN1 は}/tcfkN2 を/cf 頼って$1/ycf 上京した。 → N1 came to Tokyo from the country counting on N1^poss N2's help.
<例>わたしはおばを頼って上京した*(watasiwa ob awo tayotte joukyousita)*。"I came to Tokyo from the country counting on my aunt's help.

○(2) /y</tkN1 は>/tcfkNP2 を/cf 頼って/ycf 上京した。→<I|N1> came to town, looking to NP2 for assistance.
<例> った一人の知人を頼って上京した*(tatta hit orinochijinnwo tayotte joukyousita)*。"I came to town, looking to an only friend for assistance."

○(3) y</tkN1 は>/tcfkN2 を/cf 頼って</ycfN3 は>/cf (V4 kako |ND4 をした)。→ <I|N3> N(V4|ND4) where <I|N1> could rely on <my|N1^pron^poss> N2.
<例>親類を頼って上京した*(sinnruiwo tayotte joukyou sita)*。"I went to Tokyo where I could rely on my relatives.

○(4) /y</tkN1 は>/tcfkNP2 を/cf 頼って/ycf 上京した。→<I|N1> came to town, looking to NP2 for assistance.
<例>たった一人の知人を頼って上京した*(tatta hitorino chijinnwo tayotte joukyousita)*。"I came to town, looking to an only friend for assistance."

×(5) /y</tkN1 は>/tcfkN2 を/cfV3(て|で)/ycf(V4.kako|ND4 をした)。→<I|N1> N(V4|ND4) after having V3.past <my|N1^pron^poss> N2' consent by persuasion.
<例>両親を説得して上京した*(ryousinnwo settok usite joukyousita)*。"I came to Tokyo after having obtained my parents' consent by persuasion."

As shown in this example, the number of SP candidates was narrowed down, and the ratio of semantically correct SPs significantly increased. Based on these results, *Truth Items* are promising for selecting correct SP candidates.

5. Conclusion

We proposed *an Analogical Mapping method* based on *Semantic Typology,* and we built a semantic category system for Japanese compound and complex sentences to test this method.

This system is comprised of two subsystems: one for the classifying the meaning (222 concepts) represented by the relation between two clauses and other for the classifying the meaning (740 concepts) represented by each clause.

Based on this system, semantic codes were assigned to each SP (226,800 SPs) registered in the SP-dictionary we recently developed. We ascertained that these were useful for selecting semantically correct candidates from matched SPs for input sentences.

The system of *Truth Items* developed in this study is a semantic classification system for Japanese sentences. It is an ambitious and unprecedented attempt for semantically processing of natural expressions; however it is still tentative. We are going to improve this system through various examinations and apply it to

the *AM-method.* The results will be reported in the near future.

Acknowledgements

This research is supported by funding from the Japanese Government; the Core Research for Evolutional Science and Technology (CREST) awarded by the Japan Science and Technology Corporation (JST).

References

Brown, R. D. 1999. *Adding Linguistic Knowledge to a Lexical Example-Based Translation System.* TMI 99,:22-32.

Charles Fillmore, P. Kay, L. Michaelis, and I. Sag. 2005. *Construction Grammar.* Stanford Univ. Center, for Study of Language and Information Interaction.

Hideo Teramura. 1982. *Syntax and Semantics of Japanese Vol. 1.* Kurosio Publisher.

Hiroaki Suzuki. 1996. *Analogy and Thinking.* Kyoritsu Shuppan, Japan.

Hodumi Tanaka (Eds.). 1998. *Natural Language Processing: -Fundamentals and Applications-.* Iwanami Shoten, Japan.

Jun Arita. 1987. *German Lecture Course II.* Nankodo, Japan,:48-56.

Kikuya Ichikawa. 1963. *Methodology for Creative Research (extended edition).* Sanwa Shobo, Japan.

Koichi Takeda. 1996. *SP-based Machine Translation.* the 16th COLING, Vol. 2:1155-1158.

Lakoff. 1986. *Metaphors We Live By.* translated by Shoichi Watanabe and Kazuyuki Shimotani, Taishukan Shoten.

Makoto Nagao, Sadao Kurohashi, Satoshi Sato, Satoru Ikehara, and Hiroshi Nakano. 1998. *Science of Language.* Vol. 9, Language Information Processing, Iwanami Shoten.

Makoto Nagao. 1984. *A Framework of a Mechanical Translation between Japanese and English by Analogy Principle.* In A. Eithorn and R. Barneji (Eds.). *Artificial and Human Intelligence.* North-Holland, :173-180.

Ronald W. Langacker. 1991. *Foundations of Cognitive Grammar.* Vol. 1, Vol. 2, Stanford University Press.

Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, and Yoshihiko Hayashi. 1987. *Speaker's cognition and a Multi-Level-Translating Method based on it.* Journal of the Information Processing Society of Japan, 28(12): 1269-1279.

Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, Akio Yokoo, Hiromi Nakaiwa, Kentaro Ogura, Yoshifumi Ooyama, and Yoshihiko Hayashi. 1997 *A Japanese Lexicon.* Iwanami Shoten, Japan.

Satoru Ikehara. 2001. *Meaning Comprehension Using Semantic SPs in a Large-Scale Knowledge-Base.* Proceedings of the PACLING'01, :26-35.

Satoru Ikehara, Masato Tokuhisa, Jin'ichi Murakami, Masashi Saraki, Masahiro Miyazaki, and Naoshi Ikeda. 2006. *Pattern Dictionary Development based on Non-Compositional Language Model for Japanese Compound and Complex Sentences.* ICCPOL-06, Springer LNAI: 509-519.

Satoru Ikehara, Satsuki Abe, Masato Tokuhisa, and Jin'ichi Murakami. 2004. *Japanese to English Sentence Pattern Generations for Semantically Non-Linear Complex Sentences.* Natural Language Processing, 11(3) :70-95.

Satoshi Sato. 1992. *An example-based translation and system.* COLING-91,:1259-1263.

Takashi Masuoka and Yukinori Takubo. 1992. *Fundamental Japanese Grammar.* Kurosio Publisher, Japan.

Yasuo Nakamura. 1983. *How far can we go in translation?* The Japan Times.

Bridging the gap: Thai - Thai Sign Machine Translation

Srisavakon Dangsaart
Institute for Innovation and Development of Learning Process,
Mahidol University, Bangkok, Thailand
sri_dang@hotmail.com

Kanlaya Naruedomkul
Department of Mathematics
Mahidol University, Bangkok, Thailand
scknr@mahidol.ac.th

Nick Cercone
Department of Science and Engineering
York University, Toronto, Canada
ncercone@yorku.ca

Booncharoen Sirinaovakul
Department of Computer Engineering,
King Mongkut's University of Technology
Thonburi, Bangkok, Thailand
boon@kmutt.ac.th

Abstract

We propose a multi-phase approach, Thai-Thai Sign Machine Translation (TTSMT), to translate Thai text into Thai Sign language. Thai Sign language is the language of the deaf in Thailand.

TTSMT begins the translation process by segmenting the input sentence since Thai is a non-word boundary language, converting the segmented sentence into simple sentence forms since most Thai Sign are expressed in a sequence of such form, and then generating the intermediate sign codes which link a Thai word to its corresponding Thai Sign. The most appropriate sign codes will be selected and rearranged in the spatial grammatical order for generating the Sign language with pictures. The distinction between the Thai text and Thai Sign Language in both grammar and vocabulary are concerned in each processing step to ensure the accuracy of translation.

We have developed the Thai-Thai Sign translation system based on TTSMT. The developed system was implemented and tested to translate Thai sentences used in everyday life. The test results have proved that TTSMT can provide accurate translations.

1 Introduction

Our world is fully loaded with all kinds of information; parts of it appear in text form. To fully utilize such information, the hearing impaired and the deaf should have an ability to effectively read written text. In addition, many accessibility aids for the hearing impaired and the deaf, like television closed captioning or tele-type telephones, computer interfaces, Internet information and services, assume that the users have strong language literacy skills.

Most of the hearing impaired and the deaf experience difficulties in mastering their reading skills, and are often faced with comprehension problems when they read written text. Although hearing impaired and deaf students are taught the written language, their reading skills are lower than what they are supposed to be due to their inability to hear the spoken language. As concluded in the research at the Gallaudet Research Institute (1996), "Young deaf and hearing impaired adults aged 17 and 18 had the same average reading score as average ten-year-old hearing children".

Thus for assisting the hearing impaired and the deaf to overcome such problems and to be able to communicate in a world of spoken languages, an interpreter is needed. However, having a human interpreter at their side at all times is not the answer because of the cost and the limited availability of trained personnel. Therefore, it is not surprising that a Sign Language Automated Machine Translation System is in great demand to make more information and services accessible to the hearing impaired and the deaf

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 191–199,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

on a more economical basis when text material is too complex or an interpreter is unavailable.

A number of Sign language translation systems have been developed around the world, e.g. TESSA system (Bangham and Cox, 2000) is the direct approach and restricted domain. Weather Reports Generate System (Angus and Smith, 1999) is the transfer approach and using the limited domain. ViSiCAST Translator (Safar and Marshall, 2000) is semantically driven used HPSG semantic feature structures and Discourse Representation Structures to represent the internal structure of linguistic object that is rule-based and specific domain. TEAM Project (Zhao and Kipper, 2000) is the transfer approach that uses synchronized tree adjoining grammar. ZARDOZ system (Veale and Collins, 1998) is the interlingual systems. South African Sign Language Machine Translation System (Zijl and Barker, 2003) is transfer approach and rule-based. ASL Workbench (Armond and Speers, 2001), TGT System-Polish Text into Sign Language (Suszczanska and Szmal, 2002) are the transfer approach. Spatial and Planning Models of ASL Classifier Predicates for MT and American Sign Language Generation: Multimodal NLG with Multiple linguistic Channels (Huenerfauth, 2004,2005) are the hybrid models that combined the transfer and interlingual approaches. This work focuses in particular on models for classifier predicates. An example-based approach to translating Sign language (Morrissey and Way, 2005) applied example-based methods for automatic translation base on one of the ECHO corpora which is robust for sentences already seen in training, but has problems with unseen word and phrase combinations. Morpho-Syntax Based Statistical Methods for Automatic Sign Language Translation (Stein and Bungeroth, 2006) is statistical-based.

Most of them are English-to-American Sign Language, although there have been a few other language pairs. In Thailand, Thai Sign Language Recognition Systems and Thai Sign Language dictionaries have been developed. However, no Thai into Thai Sign language automated translation system which is able to generate reliable translation results is yet available.

To bridge the communication gap between the hearing impaired and hearing worlds in Thailand, the Thai-Thai Sign Machine Translation (TTSMT) approach is proposed. TTSMT was designed to translate Thai text into Thai Sign Language.

2 Thai and Thai Sign Language: Linguistic Issues

The Sign language uses gestures instead of sound to convey the meaning. Gestures can be characterized by manual and non-manual parameters. Manual parameters include hand-shape, hand-orientation, position, and motion while non-manual parameters include posture of the upper torso, head-orientation, facial expressions, lip movement and position (Sutton-Spence and Woll, 1999; Stokoe, 1978). Although Sign language is the native language for the hearing impaired and the deaf, it is not a universal language. Regionally, different Sign languages have evolved such as the American Sign Language (ASL) used in the United States, the British Sign Language (BSL) used in the United Kingdom and the Thai Sign Language (TSL) used in Thailand.

TSL is different from the Thai language, featuring its own lexicon and grammar rules. The sentence type in the TSL is normally simple but it corresponds to a complex or compound sentence in the Thai language. Moreover some ordering of a compound sentence in the Thai language is unlike the ordering of simple sentences in the TSL as shown in Table 1. The structure of a Thai sentence contains Subject (S), Verb (V) and Object (O) in sequence which differs from the TSL. The TSL contains O, S and V respectively and there is a Classifier[1] (CL) in some positions as shown in Table 2. A Thai phrase represents the different word sequences from a TSL phrase as shown in Table 3. In addition, the gesture of verb in the TSL depends on subject and/or object whereas in the Thai language, it does not.

Table 1. Compare some sentence ordering.

No.	Thai Sentence Ordering	TSL Sentence Ordering
1	$S = S_1$ + and $+S_2$	$S = S_1 + S_2$
2	$S = S_1$ + or $+S_2$	$S = S_1 + S_2$
3	$S = S_1$ + because $+S_2$	$S = S_2 + S_1$

Table 2. Some different syntax.

No.	Thai Syntax	TSL Syntax
Active voice		
1	S+V	S+V
2	S+Neg+V	S+V+Neg
3	S+V+O	O+CL+S+V
4	S+Neg+V+O	O+CL+S+V+Neg

[1] Classifier is the relationship between the symbols

No.	Thai Syntax	TSL Syntax
5	S+V+diO+indO	indO+CL+diO+CL+S+V
6	S+Neg+V+diO+indO	indO+CL+diO+CL+S+V +Neg
Passive voice		
7	O+S+V	O+CL+S+V
8	O+Neg+S+V	O+CL+S+V+Neg

Table 3. Compare some word sequence in a phrase.

No.	Thai Phrase Ordering	TSL Phrase Ordering
1	VP = Neg+V	VP = V+Neg
2	NP = Neg+(Adj,Adv)	NP = (Adj, Adv)+Neg
3	PP = P+NP	PP = NP+P
4	NP = NP_1+P+NP_2	NP = NP_2+CL+NP_1+P
5	NP = NP_1+P_1+ NP_2+P_2+NP_3	NP = NP_3+CL+NP_2+ P_2+CL+NP_1+P_1

3 TTSMT Architecture

The architecture of Thai-Thai Sign Machine Translation (TTSMT), an approach to translate Thai Text into Thai Sign, is illustrated in Figure 1. TTSMT comprises five modules: Sentence Treatment, Word Treatment, Sign-Code Selection, Sign-Code Ordering and Image Mapping.

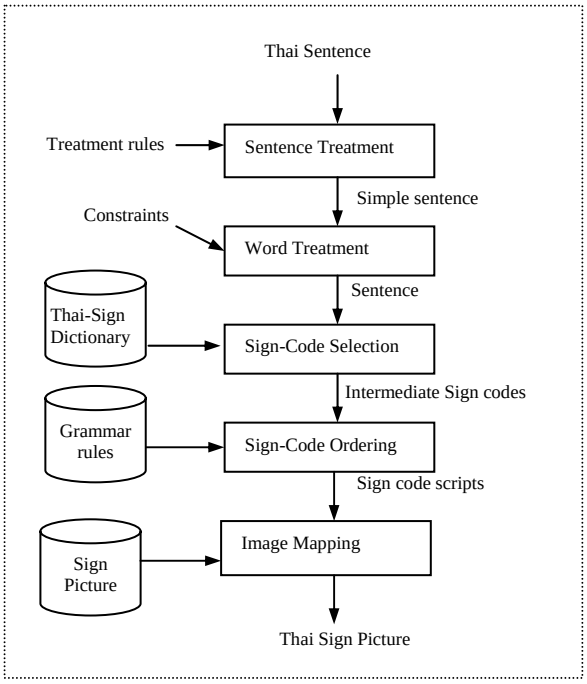


Figure 1. TTSMT architecture.

3.1 Sentence Treatment

The first module consists of two steps*: Word Segmentation* and *Sentence Segmentation*. The *Word Segmentation* step is required since the Thai language has no word boundary marker in its writing system. An automatic Thai word segmentation tool, (SWATH), was applied to segment words in TTSMT. Table 4 shows an example of word segmentation.

The treatment rules are utilized, since most TSL are not expressed in a complex or compound form. Therefore, any input string which contains a conjunction, e.g. "และ (and)", "หรือ (or)", "แต่ (but)", "เพราะ (because)" will be converted into a sequence of simple sentences in the *Sentence Segmentation* step. Table 5 illustrates an example of a segmented sentence.

Table 4. Example of a segmented word.

Segmented Input	Segmented Word
พ่อกินก๋วยเตี๋ยวแต่แม่กินข้าว (The father eats noodle, but the mother eats rice)	พ่อ-กิน-ก๋วยเตี๋ยว-แต่-แม่-กิน-ข้าว (The father eats noodle, but the mother eats rice)
[พ่อ-father],[กิน-eat],[ก๋วยเตี๋ยว-noodle],[แต่-but],[แม่-mother], [กิน-eat],[ข้าว-rice]	

Table 5. Example of a segmented sentence.

Segmented Input	Segmented Sentence
พ่อ-กิน-ก๋วยเตี๋ยว-แต่-แม่-กิน-ข้าว (The father eats noodle, but the mother eats rice)	1. (พ่อ-กิน-ก๋วยเตี๋ยว) (The father eats noodle) 2. (แต่) (but) 3. (แม่-กิน-ข้าว) (the mother eats rice)
[พ่อ-father],[กิน-eat],[ก๋วยเตี๋ยว-noodle],[แต่-but],[แม่-mother], [กิน-eat],[ข้าว-rice]	

3.2 Word Treatment

This module generates the intermediate sign code (ISC) from segment input, which will be used as a keyword in the *Sign-Code Ordering* module, in three steps: *Word Constraint, Word Addition, and Dictionary Lookup*. An ISC links a Thai word to its corresponding Thai Signs. Thai language constraints, TSL constraints, the Thai-Sign dictionary, the Thai dictionary and the Sign dictionaries are used in this module.

The Word Constraint is applied to simplify the structure of a Thai input sentence by removing some Thai words which are not required to be translated into the TSL due to the differences between Thai and TSL grammars. Such words include synonyms, conjunctions and units of countable nouns[2]. If there are synonyms in the sentence and one locates directly next to the

[2] In this research, the word "classifier" is avoided to be used as a unit of countable nouns.

others in the sentence, only the first word is kept while the others are removed. In Table 6, the word "*มากมาย* (many)" is a synonym of the word "*เยอะแยะ* (many)". Therefore, the word "*เยอะแยะ* (many)" is removed.

Table 6. Example of word constrain.

Segmented Sentences	Constrained Phrases
เด็ก-มี-ตุ๊กตา-*มากมาย*-*เยอะแยะ* (A children has many many dolls)	เด็ก-มี-ตุ๊กตา-*มากมาย* (A children has many dolls)
[เด็ก-children],[มี-has],[ตุ๊กตา-doll],[*มากมาย*-many],[*เยอะแยะ*-many]	

After applying all related constraints, any word needed to complete the string (according to the TSL grammar) and to retain the meaning of the input sentence will be added in the *Word Addition* step. In the following example (Table 7), the word "บ้าน (home)" was added to specify the place where "I" and "grandma" live in.

Table 7. Example of Word Addition.

Constrained Phrases	Word Added Phrases
ฉัน-อาศัยอยู่-กับ-ยาย (I live with grandma)	ฉัน-อาศัยอยู่-*บ้าน*-กับ-ยาย (I live at *home* with grandma)
[ฉัน-I],[อาศัยอยู่-live],[กับ-with],[ยาย-grandma],[บ้าน-home]	

In *Dictionary Lookup*, each input word will be mapped to all possible corresponding ISC. Some Thai words may correspond to more than one ISC, e.g. "กิน (eat)" and "เดิน (walk)" correspond to at least seven and three different ISCs respectively (Table 8), these results in different ISCs from the input sentence. The most appropriate ISC will be selected in the Sign-Code Selection.

Table 8. Examples of words and its corresponding to ISCs.

Word	Possible Corresponding ISC
กิน	- ISC of human-eat-rice
(eat)	- ISC of human-eat- noodle
	- ISC of mammals-four legs-eat (except elephant)
	- ISC of elephant-eat
	- ISC of birds-eat
	- ISC of aquatic-birds-eat
	- ISC of poultry-eat
เดิน	- ISC of human-walk
(walk)	- ISC of mammals-four-legs-walk
	- ISC of animal-two-legs-walk

3.3 Sign-Code Selection

The ISC's codes were created by combined some advantages of *Word Association Number* (WordAsso) (Naruedomkul and Cercone, 1999) with our new idea. The WordAsso is adapted from the concept classification, so words are classified into groups. The ISC's code property keeps the relation of ISC that can determine the involvement with other ISC codes.

The most appropriate ISC of each input word is selected in this module (if there are more than one) based on the *semantic* and *syntactic* relationship.

The Semantic relationship refers to the relation between the words in close proximity which can be determined by their context. If there are more than one ISC correspond with one input word. To select the correct ISC, each ISC's code will be examined with the left and right ISC's codes to identify which ISC's code match with the left and right ISC's codes. For example, the word "ข้าว (rice)" in the sentence "คนกินข้าว (human eats rice)" refers to "เมล็ดข้าว (rice grain)" while it refers to "ต้นข้าว (rice plant)" in the sentence "ข้าวอยู่ในนา (rice is in a field of rice)". Therefore, different ISCs will be selected for each "ข้าว (rice)" which corresponds with the semantic.

Selecting the sign gesture for a verb is quite complicated, since it must agree with subject and /or object and voice of sentences (active or passive voice).

The syntactic relationship refers to the agreement of verb, subject and object gestures that determined by the conformance of *verb* with *subject* or *verb with both subject and object*. If a verb word corresponds with many ISCs. The ISC's codes are considered with the subject and/or ISC's code of object which ISC's code relate with subject and object. For example, the verb "กิน (eat)" in the sentence "คนกินข้าว (human eats rice)" and in the sentence "คนกินก๋วยเตี๋ยว (human eats noodle)" were mapped to different ISCs since the objects of each sentence are different. In the sentence "ไก่กินข้าว (A hen eats rice)" and "หมากินข้าว (A dog eats rice)", the verb "กิน (eat)" is mapped to another ISC according to its subject. For active voice and passive voice, both sentence "ไก่กินหนอน (A hen eats worm)" and "หนอนถูกไก่กิน (The worm was eaten by a hen)", the hen eating gesture (peak) is selected, not worm eating gesture as shown in Table 9.

Table 9. Examples of selected ISC.

Phrase	Selected ISC	Sign (verb only)
Verbs depend on both subject and object		
คนกินข้าว (A human eats rice)	คน-คนกินข้าว-ข้าว (human- use a spoon to eat - rice)	spoon
คนกินก๋วยเตี๋ยว (A human eats noo-dle)	คน-คนกินก๋วยเตี๋ยว-ก๋วยเตี๋ยว (human- use a chopstick to eat- noodle)	chopstick
[คน-human],[กิน-eat],[ข้าว-rice], [ก๋วยเตี๋ยว-noodle], [คนกินข้าว-use a spoon to eat],[คนกินก๋วยเตี๋ยว-use a chopstick to eat]		
Verbs depend on subject		
ไก่กินข้าว (A hen eats rice)	ไก่-จิก-ข้าว (hen- peck -rice)	peck
หมากินข้าว (A dog eats rice)	หมา-เลีย-ข้าว (dog- lap -rice)	lap
[ไก่-hen],[กิน-eat],[ข้าว-rice],[จิก-peck][หมา-dog],[เลีย-lap]		
Active voice and passive voice		
ไก่กินหนอน (A hen eats worm)	ไก่-จิก-หนอน (A hen pecks worm)	peck
หนอนถูกไก่กิน (A worm is eaten by hen)	หนอน-ไก่-จิก (worm-hen-peaks)	peck
[ไก่-hen],[กิน-eat],[หนอน-worm],[จิก-peck]		

3.4 Sign-Code Ordering

Owing to the structures of the TSL phrases, sentences and compound sentences are different from the Thai language. Therefore, the ISCs selected in the previous step must be rearranged. To perform the correct grammatical ordering, three categories of word ordering rules were created from the syntactic level to order the intermediate sign codes (ISC) into sign code script (SCS). First, the phrase structure rules line up words in a phrase. Second, the sentence structure rules order phrases in a sentence, and the multi-sentence structure rules rearrange sentences in a compound sentence. Those rules correspond with the TSL grammar as shown in Table 10.

Table 10. Some structure rules.

No.	TSL Phrase Structure Rules
1	Neg = (V, Adj, Adv) + Neg
2	P1 = NP + P
3	P2 = NP_2+ CL+NP_1+ P
4	P3 = NP_3+ CL+ NP_2+ P_2+CL+ NP_1+ P_1
	TSL Sentence Structure Rules
5	V1 = S + V
6	V2 = S + V + Neg
7	V3 = O + CL + S + V
8	V4 = O + CL + S + V + Neg
9	V5 = indO + CL+ diO + CL+ S + V
	TSL Sentence Ordering rule
10	เพราะ [Because] = S_2 + S_1

Table 10, row 1 shows the rule for ordering a negation. Row 2 represents a preposition phrase (PP) ordering rule. Row 3 presents the rule for ordering a noun phrase (NP) and a PP while the rule for ordering NPs and PPs is in row 4. Row 5 is a intransitive verb ordering rule. Row 6 and 8 display the rule for ordering a negative sentence. Row 7 represents the rule for ordering a transitive verb while the rule in row 9 is the ditransitive verb. The rule for ordering a compound sentence indicates that in row 10.

Sign-Code Ordering rearranges the ISC string in TSL grammatical order and can order accurately both active voice and passive voice sentences. This module consists of two steps: *Ordering* and *Sentence integration steps*.

The Ordering step orders the ISC to SCS according to the TSL grammatical order. There are two steps in the Ordering step: extraction and ordering steps.

The Extraction step composes of searching and extracting processes. After the ISC was selected successfully, the searching process traverses each code in the ISC to search the head word (*Verb*, *Prep* and *Neg*). The head word is identified as a root of tree. The extracting process extracts the ISC to the sub- ISCs (NP_{ln}, NP_{rn}, *R*) as shown in Figure 2.

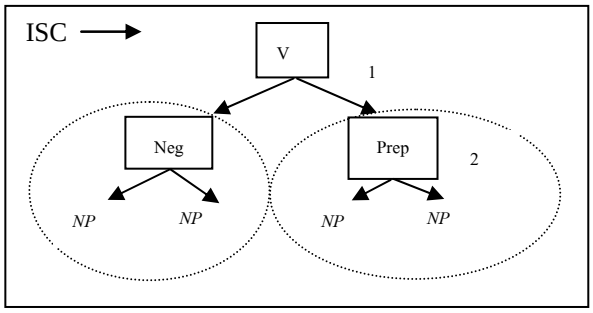


Figure 2. Example of extraction.

The Ordering step is an operation to order the ISC to the SCS. There are three processes in this step: mapping, ordering and concatenating. The mapping process maps *NP* from the lowest level of the left node to the right node groups with the

word ordering rule by considering the word category of the root of a sub-ISC. The ordering process rearranges the nodes in the *NP* which corresponds to the word ordering rule. After rearrange the word order, the concatenating process joins *NP* from the lower to the higher levels as shown in Table 11.

Table 11. Examples of Sign code ordering.

Thai sentence = แมว-ไม่-น่ารัก-กิน-ปลา-เล็ก-ใน-จาน-ใหญ่
(The cat is not beautiful eat small fish on the big dish)
ISC = {[แมว-cat],[ไม่-not],[น่ารัก-beautiful],[กิน-eat],[ปลา-fish], [เล็ก-small],[ใน-on],[จาน-dish],[ใหญ่-big]}
The extraction step:
Level 1
ISC = {[แมว-cat],[ไม่-not],[น่ารัก-beautiful],[กิน-eat],[ปลา-fish], [เล็ก-small],[ใน-on],[จาน-dish],[ใหญ่-big]}
R = {[กิน-eat]}
NP_{l1} = {[แมว-cat],[ไม่-not],[น่ารัก-beautiful]}
NP_{r1} = {[ปลา-fish],[เล็ก-small],[ใน-on],[จาน-dish],[ใหญ่-big]}
Level2L
NP= {[แมว-cat],[ไม่-not],[น่ารัก-beautiful]}
R = {[ไม่-not]}
NP_{ll2} = {[แมว-cat]}
NP_{lr2} = {[น่ารัก-beautiful]}
Level2R
NP= {[ปลา-fish],[เล็ก-small],[ใน-on],[จาน-dish],[ใหญ่-big]}
R = {[ใน-on]}
NP_{rl2} = {[ปลา-fish],[เล็ก-small]}
NP_{rr2} = {[จาน-dish],[ใหญ่-big]}
The reordering step:
Level2R
NP= {[ปลา-fish],[เล็ก-small]},{[ใน-on]},{[จาน-dish],[ใหญ่-big]}
Mapped *NP* with the ordering rule $NP = NP_2+CL+ NP_1+ P$
Reorder *SCS* = {[จาน-dish],[ใหญ่-big],CL,[ปลา-fish], [เล็ก-small],[ใน-on]}
Level2L
NP= {[แมว-cat],[ไม่-not],[น่ารัก-beautiful]}
Mapped *NP* with the ordering rule $Neg = NP_1+ NP_2+ Neg$
Reorder *SCS* = {[แมว-cat],[น่ารัก-beautiful],[ไม่-not]}
Level1
NP= {[แมว-cat],[น่ารัก-beautiful],[ไม่-not]},{[กิน-eat]}, {[จาน-dish],[ใหญ่-big],CL,[ปลา-fish],[เล็ก-small],[ใน-on]}
Mapped *NP* with the ordering rule $V = O + CL+S + V$
Reorder *SCS* = {[จาน-dish],[ใหญ่-big],CL,[ปลา-fish],[เล็ก-small], [ใน-on],CL,[แมว-cat],[น่ารัก-beautiful],[ไม่-not] [กิน-eat]}
Generated sign code script
SCS = {[จาน-dish],[ใหญ่-big],CL,[ปลา-fish],[เล็ก-small], [ใน-on],CL,[แมว-cat],[น่ารัก-beautiful],[ไม่-not],[กิน-eat]}

Figure 2 and Table 11 show the extracting and ordering of the ISC "แมว-ไม่-น่ารัก-กิน-ปลา-เล็ก-ใน-จาน-ใหญ่ (The cat is not beautiful eat small fish on the big dish)".

The last step, *Sentence Integration*, is activated only if the input sentence was divided into sub-sentences. The sentence integration combines all sub-sentences together. As previously stated ISCs have been ordered to SCSs separately. To complete the ordering, all of SCSs are integrated into one SCS. There are two processes in this step: mapping and integration processes. The mapping process matches the conjunction with the multi-sentence rule to integrate the SCSs. The integration process combines the SCSs which correspond to the rule as shown in Table 12.

3.5 Image Mapping

The final module generates TSL by mapping a list of SCS with the pictures of the TSL.

Table 12. Examples of Sentence Integration.

Thai Sentence	Sentence Integration
1. พ่อ-ได้-เงิน-มาก (The father gets a lot of money) 2. เพราะ (because) 3. พ่อ-ทำงาน-หนัก (The father works hard)	[พ่อ-father],[ทำงาน-work], [หนัก-hard],CL,[เงิน-money], [มาก-a lot of],[พ่อ-father], [ได้-get]
[พ่อ-father],[ได้-get],[เงิน-money],[มาก-a lot of],[เพราะ-because], [ทำงาน-work],[หนัก-hard]	

4 Knowledge-Based Representation

The essential Thai and TSL knowledge-bases required for TTSMT include: Treatment, Constraint, Dictionary, Grammar and Lexicon. The knowledge-bases are developed separately for easy modification, extendibility and management.

Treatment is the set of specific characteristics of the Thai language that is not available in the TSL e.g. "และ (and)", "หรือ (or)", "แต่ (but)", "เพราะ (because)".

Constraints are the different characteristics between the Thai language and the TSL. There are two sets of constraints: Thai language and TSL constraints.

- Thai language constraints are used to simplify the structure of the Thai language and to narrow the scope of possible TSL words which correspond to each Thai language word such as the synonym "สวย (beautiful)", "น่ารัก (pretty)", "มีเสน่ห์ (charming)", conjunctions "และ (and)" and units of countable nouns "อัน (aun)", "ชิ้น (chein)".

- TSL constraints are required to retain the meaning of the Thai language and to make them grammatically correct. For example the words "พักอยู่(stay)", "อาศัย (live)", "อยู่อาศัย (reside)" are the intransitive verb in the Thai language, but these

must remain in a place in the TSL. To specify the place, the word "บ้าน (home)" must be added.

Dictionary, There are three sets of dictionary: Thai dictionary, Sign dictionary and Thai-Sign dictionary. Each entry in dictionary contains semantic and syntactic categories.

Grammars and Lexicons of the Thai language and the TSL have been developed base on the HPSG (Pollard and Sag, 1994). This grammar formalism is well suited to the TSL because it operates on feature structures allowing access and modification of various levels of linguistic specification. Each lexical entry contains morphological, syntactic and semantic information.

5 Experiment

The Thai to Thai Sign MT system has been developed using TTSMT approach and run under SWI Prolog 5.2. The grammar using in this system was developed using Attribute Logic Engine version 3.2.1. With the proposed design, TTSMT does not need a human operator to intervene between pipeline stages to correct the linguistic analysis prior to final surface realization.

The developed system was used to translate phrases/sentences. Some translation results are shown in Table 13.

Table 13. Examples of translation results.

Thai Sentence	Intermediate Sign Code	Sign Code Script
ลูกเป็ด-ไม่-น่ารัก (The duckling is not cute.)	[ลูกเป็ด-duckling], [ไม่-not], [น่ารัก-cute]	[ลูกเป็ด-duckling], [น่ารัก-cute], [ไม่-not]
บ้าน-ไม่-พัง (The home does not fall down)	[บ้าน-home],[ไม่-not], [พัง-fall down]	[บ้าน-home], [พัง-fall down], [ไม่-not]
หมาป่า-เจ้าเล่ห์-เห็น-หมูน้อย-ใน-บ้าน (The sly wolf see the little pig in the home)	[หมาป่า-wolf], [เจ้าเล่ห์-sly], [เห็น-see], [หมูน้อย-little pig], [ใน-in],[บ้าน-home]	[บ้าน-home],CL, [หมูน้อย-little pig], [ใน-in],CL, [หมาป่า-wolf], [เจ้าเล่ห์-sly], [เห็น-see]
ชาวนาใจดีให้ฟางทั้งหมดกับหมูน้อย (The kind farmer give all straw to the little pig)	[ชาวนา-farmer], [ใจดี-kind],[ให้-give], [ฟาง-straw], [ทั้งหมด-all],[กับ-to], [หมูน้อย-little pig]	[หมูน้อย-little pig], CL,[ฟาง-straw], [ทั้งหมด-all],CL, [ชาวนา-farmer], [ใจดี-kind], [ให้-give]

Table 13, the negation in the phrase is translated in row 1. Row 2 shows the translation of the negation in the sentence. Row 3 represents the translation of the preposition in the sentence. The translation of ditransitive verb is in row 4.

6 TTSMT Evaluation

TTSMT evaluation was restricted to test the translated processes without considering the potential environments. The 297 test sample sentences /phrases were collected from different sources: textbooks (affirmative, negative, interrogative, order, compound sentences, active and passive voice), the bedtime story (The three little pigs), cartoons, newspapers and the public labels.

The accuracy, precision, recall and F-measure were used to determine the translation in terms of intelligibility and fidelity by five testers including two linguistic teachers (hearing and deaf), two deaf students and one Thai sign interpreter. All testers are from Ratchasuda College, a college for persons with disabilities in Bangkok, Thailand. Here, the intelligibility and fidelity are evaluated from the following 7 measures. The evaluation results are shown in Table 14.

1 = correct grammar,
2 = correct word usage,
3 = inappropriate word usage,
4 = incorrectly translated word,
5 = correct CL,
6 = convey the original meaning,
7 = convey the different meaning but can understand the original meaning.

Table14. The intelligibility and fidelity evaluation results.

Measurement							Accu-racy (%)	Preci-sion (%)	Recall (%)	F (%)
1	2	3	4	5	6	7				
*	*			*	*		94	100	94	96.9
*		*		*	*		95	100	95	97.4
*			*			*	94	100	94	96.9
	*			*		*	93	100	93	96.3
*	*				*		94	100	94	96.9

In Table 14, row 1 displays the grammar, word usage and CL that are correct and convey the original meaning. Correct grammar, inappropriate word usage, correct CL and convey the original meaning are shown in row 2. Row 3 presents correct grammar, incorrectly translated

word, convey the different meaning but can understand the original meaning. In the same way correct word usage, correct CL, convey the different meaning but can understand the original meaning are represented in row 4. Correct grammar, correct word usage and convey the original meaning are in row 5.

The first step, TTSMT can be augmented into the Thai-Thai Sign educational application by mapping the SCS with pictures of TSL gestures. The application was developed to satisfy the needs of users, so the user interface was designed to support users from the beginning level and deaf, hearing impaired students whose first language is the TSL as shown in Figure 3.

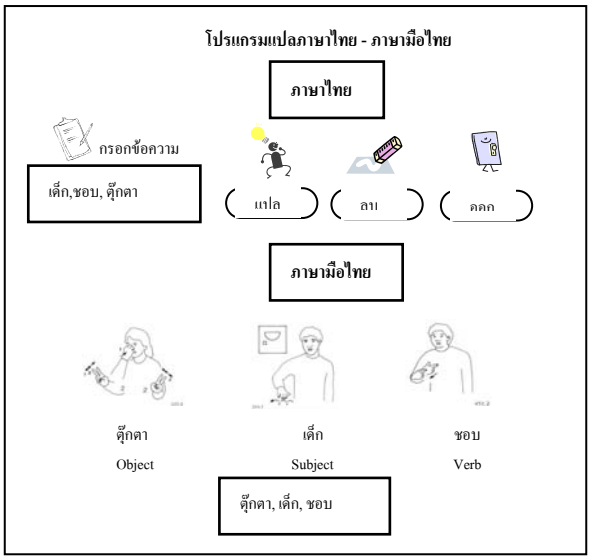


Figure 3. TTSMT user interface.

The questionnaires were used to collect data to evaluate users' satisfaction in terms of simple and convenient, good advantage and need of user. The users' satisfaction evaluation results as shown in Table 15.

There are five levels of satisfaction: 5 means excellent; 4 means good; 3 means fair; 2 means poor; 1 means very poor.

Table 15. Users satisfaction evaluation.

	Simple Convenient	Good advantage	Need of user
Mean	4.03	4.12	4.07
S.D.	0.66	0.65	0.64
Fair (%)	20.2	16.1	17.2
Good (%)	56.3	55.8	58.6
Excellent (%)	23.5	28.1	24.2

The results in Table 15 show that users satisfy in Thai-Thai Sign education application.

7 Concluding Remarks

We present TTSMT, an alternative approach to translate the Thai language into the Thai Sign language. TTSMT is able to generate an accurate translation result since it takes into account the differences between the Thai language and the TSL in terms of both lexicons and syntax. TTSMT exploits the advantage of semantic and syntactic relationship between words in close proximity to generate an appropriate list of sign code script which best represents input Thai sentences.

In our initial implementation, TTSMT was applied to develop an automated translation system for assisting hearing impaired and deaf people to enhance their abilities in learning, to read and to communicate with other people.

In addition, with some educational applications, this system can turn into a Thai Sign Language (tutor) training tool for students (both hearing and hearing impaired) to learn the Thai language and the TSL faster and easier.

References

Angus B. and Smith G. 1999. English to American Sign Language machine translation of weather reports. *Proceedings of the second High desert Student Conference in Linguistics, Albuquerque, NM.*

Armond D. and Speers L. 2001. *Representation of American Sign Language for Machine Translation*. PhD dissertation, Department of linguistics, Georgetown University.

Bangham J. A. and Cox S. J. 2000. Signing for the deaf using virtual humans. *In IEEE Colloquium on speech and Language Processing for the Disabled and Elderly People, London, England.*

Gallaudet Research Institute. 1996. *Stanford Achievement test,* 9th Edition, Form S, Norms Booklet for Deaf and Hard-of-Hearing Student. Gallaudet University, Washington.

Huenerfauth M. 2004. Spatial and Planning Models of ASL Classifier Predicates for Machine Translation. *In The 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2004)* Baltimore, MD., pp.65–74.

Huenerfauth M. 2005. American Sign Language Generation: Multimodal NLG with Multiple Linguistic Channels. *In Proceedings of the ACL Student Re-*

search Workshop (ACL 2005) Ann Arbor,MI., pp.37–42.

Morrissey S. and Way A. 2005. An example-based approach to translating sign language. *In Workshop Example-Based Machine Translation (MT X–05)*, pp. 109–116, Phuket, Thailand, September.

Naruedomkul K. and Cercone N. 1999. The Role for Word Association Numbers in Machine Translation. *Proceedings of the Conference Pacific Association for Computational Linguistics (PACLING'99)*, Waterloo, Ontario, Canada, pp.379-392.

National Association of the Deaf in Thailand. 2000. *Thai Sign Language Book*. Baphit Press: Bangkok (Thailand).

Pollard C. and Sag A. I. 1994. *Head-driven Phrase Structure Grammar*. Center for the Study of language and Information, Stanford, The University of Chicago Press, Chicago and London.

Safar E. and Marshall I. 2000. The architecture of an English-Text-to-Sign-Language translation system. *In G. Angelova, editor, Recent Advances in Natural Language Processing (RANLP)*, pp. 223-228, Tzigov Chark, Bulgaria.

Stokoe W. C. 1978. *Sign language structure*. (2nd ed.), Silver Spring, MD:Linstok Press.

Suszczanska N. and Szmal P. 2002. Translating Polish Text into Sign Language in the TGT System. *Proceeding of the 20th IASTED International Multi-Conference Applied Informatics AI 2002*, Innsbruck, Austria 2002, pp. 282-287.

Sutton-Spence R. and Woll B. 1999. *The linguistics of British Sign language, An Introduction*. University Press, Cambridge.

SWATH. Smart Word Analysis for Thai., *http://www.links.nectec.or.th/download.php*, as of 10/01/07.

Veale T. and Collins B. 1998. The challenges of cross-modal translation: English to sign language translation in the ZARDOZ system. *In Machine Translation 13*, pp. 81-106.

Zhao L. and Kipper K. 2000. A Machine Translation System from English to American Sign Language. *Association for Machine Translation in the Americans.*

Zijl L. V. and Barker D. 2003. A Machine Translation System for South African Sign Language. *In ACM SIGGRAPH Conference Proceedings Series*, Cape Town South Africa, February, 2003.

Preparing, Restructuring, and Augmenting a French Treebank: Lexicalised Parsers or Coherent Treebanks?

Natalie Schluter
NCLT, School of Computing
Dublin City University
Dublin, Ireland
nschluter@computing.dcu.ie

Josef van Genabith
NCLT, School of Computing
Dublin City University
Dublin, Ireland
josef@computing.dcu.ie

Abstract

We present the Modified French Treebank (MFT), a completely revamped French Treebank, derived from the Paris 7 Treebank (P7T), which is cleaner, more coherent, has several transformed structures, and introduces new linguistic analyses. To determine the effect of these changes, we investigate how the MFT fares in statistical parsing. Probabilistic parsers trained on the MFT training set (currently 3800 trees) already perform better than their counterparts trained on five times the P7T data (18,548 trees), providing an extreme example of the importance of data quality over quantity in statistical parsing. Moreover, regression analysis on the learning curve of parsers trained on the MFT lead to the prediction that parsers trained on the full projected 18,548 tree MFT training set will far outscore their counterparts trained on the full P7T. These analyses also show how problematic data can lead to problematic conclusions–in particular, we find that lexicalisation in the probabilistic parsing of French is probably not as crucial as was once thought (Arun and Keller (2005)).

1 Introduction

The construction of the Paris 7 Treebank (P7T) brought to fruition the first treebank available for French (Abeillé et al. (2004); Abeillé and Barrier (2004)). Its use in research, however, has proven challenging. Arun and Keller (2005), for example, observe a number of points in which the treebank should be improved or even completely structurally reorganised before any serious study can be carried out using it.

Our goal has been to create a French treebank with consistent and coherent annotation and with a comparatively low error rate, that supports efficient statistical parsing paradigms while compromising as little as possible on linguistically relevant structural information. We hope to have done this, while carrying out only the minimum number of changes to the P7T necessary to meet this goal.

The necessary correction and modification of the P7T has led to the creation of the *Modified P7T*, which we will simply call Modified French Treebank (MFT). Our research focusses on the functionally annotated subset of 9357 sentences from the P7T, and the MFT now consists of the the first half of these sentences.

Following an overview of the P7T (Section 2), we introduce the MFT via the various structural changes (Section 3), formatting and error mining (Section 4). Using statistical analysis techniques, we show that the MFT and P7T have become very different treebanks (Section 5). Finally, as a means of showing the importance of such changes in treebank-based linguistic analysis, we give results for statistical parsing in Section 6, and draw some important conclusions.

2 The Paris 7 Treebank

Work on the P7T was carried out by a research team at the Université Paris 7, under the direction of Anne Abeillé. The treebank consists of *Le Monde* newspaper article excerpts published between 1989 and 1993, written by various authors, and covering an array of topics. The full P7T contains 20,648 sentences annotated for phrase structure, (and additionally, about half with grammatical function tags) comprising 580,945 words. Table 1 gives the phrase tags of the P7T. We notice, in particular, that there is no VP, except in the cases of some participial phrases (VPpart) and infinitival phrases (VPinf).[1]

[1] The phrase VN is considered to be more of a convention, grouping together all parts of composed verbs into one unit with their clitic pronouns, as well as any modifier phrases

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 200–209,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

AP	adjectival phrase
VPinf	infinitival phrase
AdP	adverbial phrase
Srel	relative clause
COORD	coordinated phrase
Ssub	subordinated clause
NP	noun phrase
Sint	internal, inflected sentence
PP	prepositional phrase
VN	verb kernel
VPpart	participial phrase
SENT	independent sentence

Table 1: Phrase Tags of the Paris 7 Treebank

Table 2 gives the syntactic function labels used in the functionally annotated sections of the P7T. Only some clitics and those phrases which are sisters of a VN constituent carry functional annotations. This assumes that any phrase which is a sister element of VN functionally depends directly on the verb kernel; we show that this is not always the case and present a new functional annotation scheme in Section 3.5.

SUJ	subject	DE-OBJ	de (of/from) -object
OBJ	object	A-OBJ	à (to)-object
P-OBJ	preposition-object	MOD	modifier
ATS	subject attribute	ATO	object attribute

Table 2: Syntactic Function Labels of the Paris 7 Treebank

Our project focusses on the first half of the functionally annotated sentences of the treebank; there are, in total, 20 files that contain the 9357 functionally annotated sentences, and we are working with the first ten of these files. These files originally contain 4741 sentences, comprising 134,445 words.

3 Structural Changes

The MFT differs significantly from the P7T, in terms of its phrase structure as shown by the statistical tests in Section 5. Major structural changes to the original P7T trees include increased rule stratification, introduction of analyses for untreated structures, information propagation, coordination raising, the addition of missing functional tags, and the introduction of functional path tags.

occurring between these.

3.1 Rule Stratification

While maintaining a relatively flat syntactic analysis, the MFT has the property that there is one distinct head (and sometimes also one co-head) for each constituent. For example, NP, AP, and AdP constituents that have modifiers will have separate constituents for those modifiers. Figure 1 provides an example of increased stratification for AdP in Example (1).

(1) encore pas très bien
still not very well
'still not very well'[2]

3.2 Introduction of Analyses for Untreated Structures

Compared to the P7T, the MFT offers increased coverage of linguistic phenomena. 'It'-cleft constructions are an example of structures that remained untreated in the P7T annotation guidelines, and therefore received a variety of treatments throughout the P7T. Figure 2 (for Example (2)) illustrates the new analysis, inspired mainly by van der Beek (2003).

(2) C'est [...] l'URSS [...] qui se
It is [...] the USSR [...] who herself
trouve prise [...]
finds taken [...]
'It is the USSR that finds itself trapped'[3]

3.3 Information Propagation

Some constituent categories in the P7T derive terminal strings with grammatical patterns not reflected in the intervening levels of syntactic representation. VPinf, VPpart, and Srel are the three categories which were found to have this property. For instance, VPinf necessitates a VN daughter that has a V daughter which is an infinitive, and Srel necessitates a PP or NP daughter whose head is or has an argument that has a relative pronoun daughter. Such cases amount to information loss across levels of representation, thereby introducing CFG ambiguity. A parser must guess the daughters of these constituents VN, NP, and PP, in order to carry out correct annotation. This potentially leads to poor statistical parsing. We automatically propagate the required information, aug-

[2] Sentence 88, file flmf7ag1ep.cat.xml.
[3] Sentence 8151, file flmf3_08000_08499ep.xd.cat.xml.

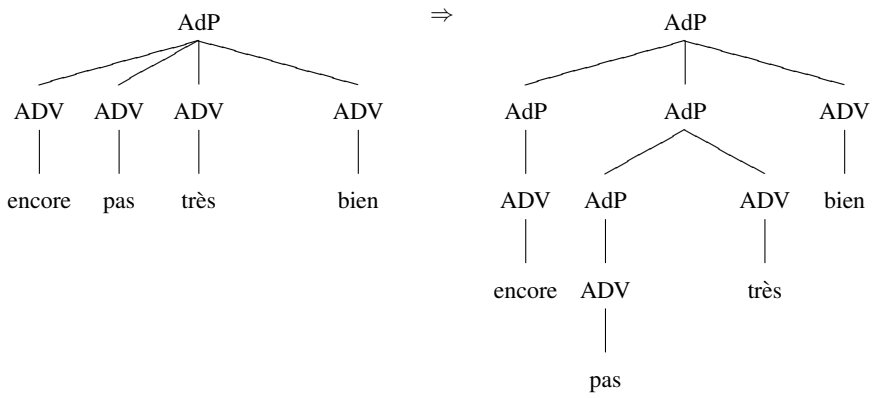


Figure 1: P7T representation (left) and MFT representation (right) of example (1).

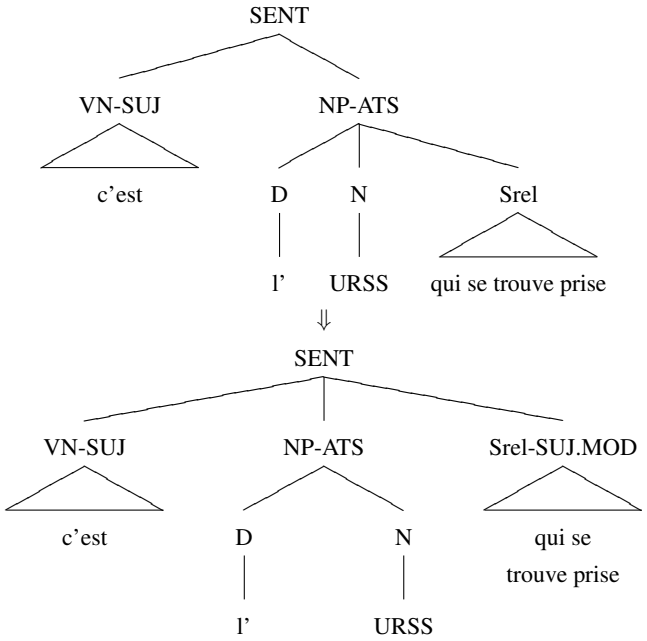


Figure 2: P7T representation (above) and MFT representation (below) of example (2).

menting the MFT with these constituent labels.[4]

- In the MFT, the part-of-speech V is separated into three different categories: Vfinite, Vinf, and Vpart, according to whether the verb is tensed, infinite, or a participial. The XML representation of the constituent VN for the MFT now has an attribute "type", which records which is the first verb POS: finite, inf, or part. Now the VPinf constituent will only have a VN constituent with type "inf", and similarly for VPpart.

- Relative pronouns in the P7T are already indicated in the "subcat" attribute. We propagate this information as "type" attributes through the dominant nodes, until the dominant node Srel is reached, thus introducing the constituent categories PPrel and NPrel.

Example (3), whose tree structure is shown in Figure (3), illustrates both these changes.

[4]Note that this is similar to the strategy suggested by Johnson (1998), but with two important differences. First, the information propagation is done here in a bottom-up fashion and, therefore, retains the central linguistic motivation behind phrase structure trees, that of constituents making up and determining a type of phrase. On the other hand, Johnson (1998) suggests a sort of information propagation in a top-down fashion—a sort of after the fact description of a phrase's context within a given tree. Second, we are not carrying out transformations to be undone after some parsing process; we are carrying out static annotation of treebank trees.

(3) [...] qui risquait de brouiller
[...] who was risking of shake-up
l'image [...]
the image [...]
'who risked messing up the image'[5]

3.4 Raised Coordination

Coordination in the P7T is represented as a sort of adjunction of a COORD phrase as a sister or daughter of the element it is to be coordinated with. This is interpreted in two different ways in the treebank, illustrated in Figure 4, making coordinated structures in the P7T highly ambiguous and inconsistent. Either of the two analyses shown are attested in the P7T, as well as a third, sometimes, for PP coordination.[6,7]

The coordination analysis adopted for the MFT is similar to that of the Penn Treebank (Bies et al. (1995)), except for one important fact: we do not get rid of the COORD constituent. Coordination has been modified to be structured as a single phrase consisting of coordinate daughters. This process of restructuring was carried out in a semi-automatic fashion. All sentences had to be hand corrected after automatic transformation, due to the ambiguity in the structures of the P7T. Generally, the goal of the transformation was to arrive at a structure such as the one in Figure 5, from those in Figure 4 (as well as from any other erroneous coordinated structures encountered).

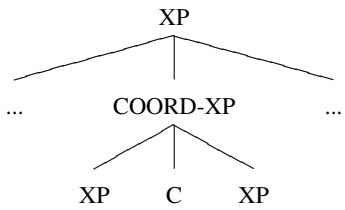


Figure 5: MFT coordination with arguments.

For like-constituent coordination, COORD XML elements now have a "type" attribute, whose value is the type of coordinated constituent (i.e., NP, AP, etc.). In Figure 5, the COORD phrase is of type XP. In addition, it is enclosed in an XP phrase along with any of its shared arguments or modifiers.

Nonconstituent coordination and unlike constituent coordination required slightly different, but similarly structured, analyses. Unlike constituent coordination was labeled with the type UC, and nonconstituent coordination with the type NC, or VP in the case of an NC that really corresponds to a VP.[8]

COORD-UC phrases may take a functional label if they are sister to a VN, whereas COORD-NC phrases do not. In NC coordination, parallel elements are enclosed in a special NC phrase, if they are not argumentally complete verbal phrases phrases (for example, argument cluster coordination). The functional roles of each of their constituents is given on the constituents themselves within the NC or Sint constituent.[9] Figure 6 illustrates a type of NC coordination for the following example.

(4) la personalité morale de la Cinq
the personality moral of the Five
disparaît, et avec elle l'autorisation
disappears, and with her the authorisation
d'émettre
of broadcast
'the moral personality of the Five is disappearing, and with it the permission to broadcast' [10]

3.5 Functional Path Tags

Approximately half of the P7T was automatically functionally annotated and hand corrected (Abeillé and Barrier (2004), cf. Section 2). In the original subsection of the P7T (before being modified and hand corrected by the present authors) the functional tag counts are as given in Table 3.

The functional annotation scheme adopted for the P7T assumed that all sisters of the VN phrase are functionally dependent on that phrase. However, this is not always the case; it-cleft construc-

[5] Sentence 8009, file flmf3_08000_08499ep.xd.cat.xml.

[6] The annotation guidelines of the P7T suggest that there is a difference in distribution, however, upon working with the P7T, one realises that this is not the case. It seems that the flatness of analyses in the trees of the P7T combined with their analysis of coordination has resulted in confused structures. Thus, for any type of constituent coordination, we possibly find either of the different structures in the P7T indicated in Figure 4.

[7] We have also found another regularly used form of coordination for PP coordination, where a PP is coordinated with the mother node of its mother node. However, we think that this is perhaps a consistent error, and not an analysis.

[8] Recall that VP is not a constituent in the P7T, and is not introduced into the MFT, except where NC would correspond to a VP.

[9] In reality, like VN, NC is not really a phrase; rather, it is a convention permitting the expression of parallel structures. We use explicity the tag "NC" to make this clear.

[10] Sentence 154, file flmfaa1ep.cat.xml.

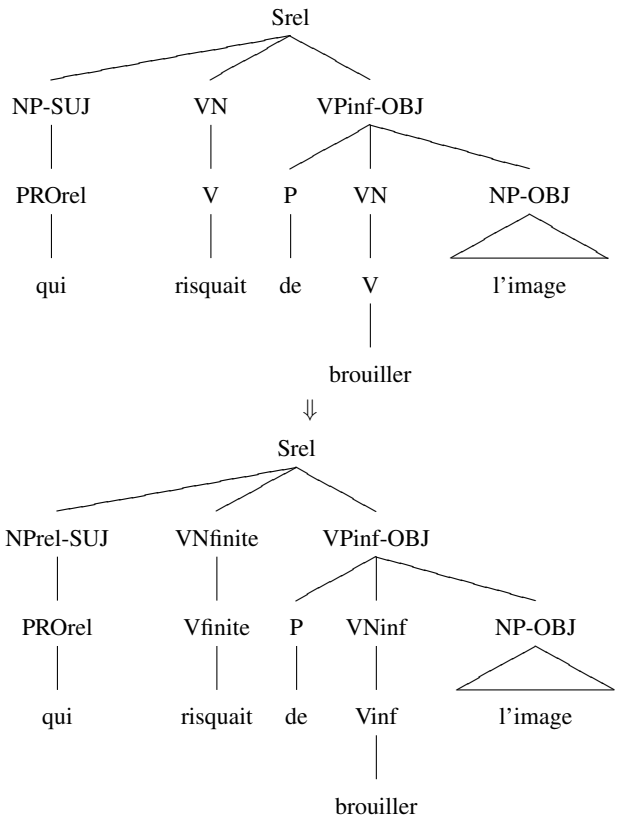


Figure 3: P7T (above) and MFT representation (below) of example (3).

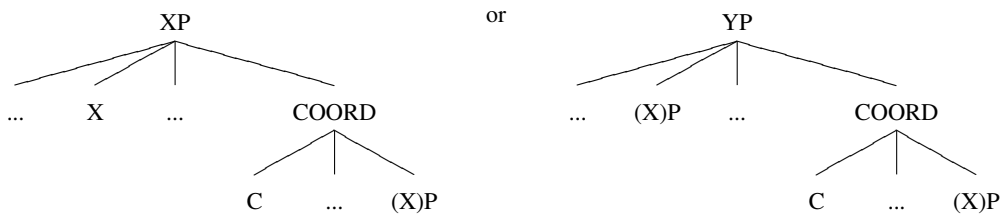


Figure 4: Coordination with Mother or with Sister Node in the P7T. The (X)P are coordinated.

Functional Tag	count	Functional Tag	count
SUJ	8036	OBJ	5949
MOD	6023	A-OBJ	833
DE-OBJ	1354	P-OBJ	913
ATS	560	ATO	104

Table 3: Original Functional Tag Counts for the Relevant P7T Subset

tions provide a first example (cf. Section 3.2). Other cases involve, for example, pronouns for DE prepositional phrases (pronouns such as *dont* or *en*) and daughters of NC. Inspired by the functional paths in the LFG framework[11], we assign new path functions, as illustrated in Figure (2), where the Srel constituent takes the functional path tag SUJ.MOD, representing the fact that Srel has the function MOD, and is dependent on the constituent whose function is SUJ.

We note, in addition, that much of the functional annotation was missing in the functionally annotated subset of the P7T; only 23,772 functional tags were found in the relevant subsection of the P7T. In contrast, the MFT contains 30,399 functional tags. Table 4 presents the MFT counts of the new functional path tags.

4 Formatting and Error Mining

In order to be usable by software, and before any restructuring of the P7T could take place, we carried out an extensive clean-up of the original P7T

[11] See, for example, Dalrymple (2001).

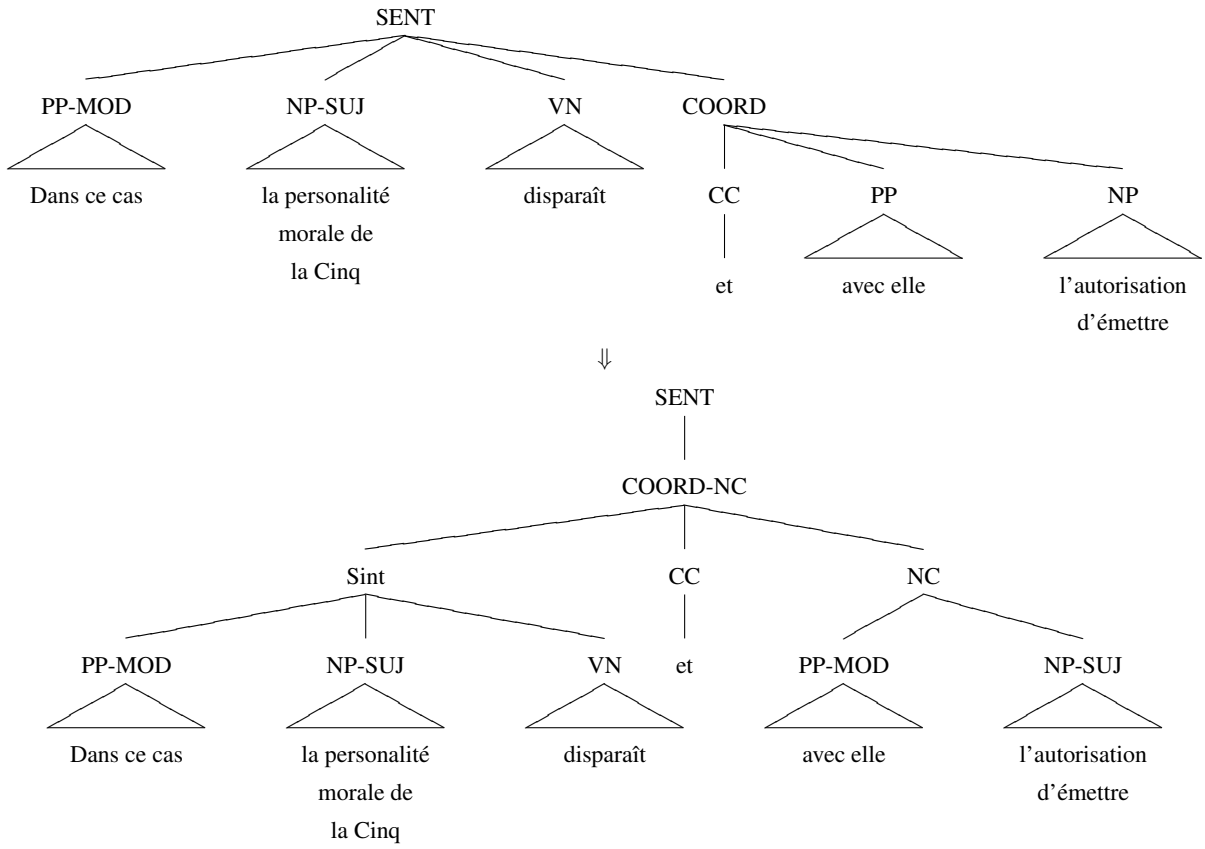


Figure 6: P7T (above) and MFT (below) representation of example (4).

Functional Tag	Count
SUJ	7969
OBJ	6667
MOD	10615
A-OBJ	1432
DE-OBJ	956
ATS	1470
SUJ.MOD	158
P-OBJ	1022
ATO	126
A-OBJ.OBJ	1
ATS.MOD	14
DE-OBJ.OBJ	1
OBJ.MOD	38
OBJ.DE-OBJ	1
OBJ.OBJ	3
SUJ.A-OBJ	1
DE-OBJ.OBJ.MOD	2
OBJ.A-OBJ	2
SUJ.DE-OBJ	1
A-OBJ.OBJ.MOD	1

Table 4: MFT Counts of Functional Path Tags

formatting. This involved, for example, reinserting missing part-of-speech tags, and repairing the XML formatting.[12]

Following the reformatting and restructuring of the treebank, a phase of general error mining and correction was undertaken to reduce any noise that we had introduced into the new MFT version of the treebank, and to try to catch any important errors that we had as yet left untreated or that we had missed. Error mining has been shown to improve the results of even very robust techniques for comparatively large corpora (Dickinson and Meurers (2005, 2003a,b)).

This phase has been carried out semi-automatically, in three steps. The first step simply involved automatically extracting a CFG grammar from the treebank, and verifying manually that the productions were consistent with P7T and MFT annotation guidelines, correcting any deviations. The next two steps consisted of applying error-mining software created under the Decca project (Dickinson and Meurers (2005)). This involved applying software for the detection

[12]For example, in the whole of the functionally annotated section of the P7T, we found 5 empty SENT constituents, 3 cases of word-forms floating outside of their XML elements, 15 misformatted lemmas, 24 missing parts-of-speech for words not belonging to a multi-word expression, 16,222 missing parts-of-speech for words belonging to a multi-word expression, 18 misused attributes, etc.

of part-of-speech variations and constituent-string relation variations, examining non-fringe results, and manually correcting any detected erroneous annotations.[13]

5 Comparative Statistics

The comparative counts of tokens and types of CFG rules for the relevant subset of sentences, given a certain left-hand side, is presented in Table 5.[14] We observe that in all instances (except for AdP[15]), the number of tokens has increased from P7T to MFT, whereas, except for Sint, the number of types has decreased. In addition, we have used a 2-sample χ^2 test for equality to show that all type-token proportions have significantly decreased, as shown in the last column of Table 5 (ranging from the largest P-value 2.546E-02 to the smallest 2.2E-16). The differences reflect the consistency and comparative simplicity of the MFT with respect to the P7T.

Left Side	P7T types/tokens	MFT types/tokens	p-value
SENT	1476/4741	1114/4739	2.2E-16
AP	93/5506	64/8440	5.428E-07
AdP	37/290	44/5755	2.2E-16
NP (or NPrel)	1086/34747	690/38036	2.2E-16
PP (or PPrel)	129/19071	60/19930	1.416E-07
VPinf	300/2940	221/3047	6.229E-05
VPpart	249/2009	160/2115	2.838E-07
Srel	302/1567	233/1590	6.475E-04
Ssub	361/1426	284/1513	2.235E-05
Sint	191/597	273/1024	2.546E-02

Table 5: Productions of the P7T versus the MFT

6 Parsing Results and Regression Analysis

Arun and Keller (2005) report parsing results on the P7T. Post-publication, Arun discovered (personal communication) that the results reported in these publications were erroneously obtained; Arun and Keller (2005) mistakenly discarded over half of the treebank trees, believing that the contracted words were XML errors. Their new results for sentences of length ≤ 40 words were given in their presentation at ACL, and are reported in Table 6.[16]

Parser and Mode	LR	LP	f-score
BitPar (own POS tagging)	64.49	64.36	64.42
BitPar (perfect tagging mode)	67.78	67.07	67.42
Bikel (own POS tagging)	79.94	79.36	79.65
Bikel (unknown POS supplied)	80.79	80.23	80.50

Table 6: Arun and Keller's P7T parsing results ($\leq$ 40 words)

Arun and Keller present results for BitPar (Schmid (2004)), as well as for several modifications made to Bikel's parser (Bikel (2002)). What they term as "Collins Model 2" is essentially Bikel's parser without any of the added modifications; results from this model applied to the best of the Arun and Keller (2005)'s transformations of the P7T (contracted compounds and raised coordination[17]) will serve as a baseline for comparison with our results here.

Upon finding that Bikel's parser outperforms BitPar when trained on the P7T by over 15%, Arun and Keller concluded that French, like English but unlike German, parses best in a lexicalised statistical parsing framework, leading to the conjecture that word order, and not flatness of annotation, is crucial for lexicalisation. By contrast, parsing results with the MFT lead to a less extreme conclusion, and provide further evidence that a coherent and well-structured treebank leads to better parsing results.

Experiments were repeated on the MFT using both BitPar and Bikel's parser. The MFT was randomly subdivided into a training set (3800 sentences), development set (509 sentences) and a test set (430 sentences). Our training set roughly corresponds (in quantity) to only 20.5% of the training data used by Arun and Keller in their most recent experiments (18,548 sentences), yet our re-

[13] For example, Decca POS software detects 28 7-gram variations of which 15 are non-fringe. The non-fringe variations were examined for errors. The same softwares detects only 11 7-gram variations in the MFT, of which 5 are non-fringe.

[14] COORD and VN, and any new constituents added to the MFT are not mentioned for reasons of incomparability. Also note that these rule counts make abstraction of any punctuation or functional tagging.

[15] Observe that the AdP phrase in the original P7T was comparatively rarely employed.

[16] The ACL slides presenting these new results may be obtained at http://homepages.inf.ed.ac.uk/s0343799/acl2005slides.pdf .

[17] As with Arun and Keller's work, we contracted compounds for our experiments. Their method of raising coordination is completely different from the one discussed here. See (Arun and Keller (2005)) for details.

sults show improvements on results using the P7T. The results for the MFT are shown in Table 7.

Parser and Mode	LR	LP	f-score
BitPar (own POS tagging)	70.66	70.62	70.64
BitPar (perfect tagging mode)	78.07	77.36	77.71
Bikel (own POS tagging)	79.76	80.13	79.95
Bikel (unknown POS supplied)	83.09	83.31	83.20
Bikel (perfect tagging mode)	84.62	84.69	84.66

Table 7: MFT parsing results ($\leq$ 40 words)

BitPar trained on the MFT outscores across the board its scores when trained on more than five times the amount of data from the P7T. On sentences of length less than 40 words, BitPar trained on the MFT scores 6.22% better, and in perfect tagging mode, BitPar scores 10.29% better than when trained on the substantially larger training set from the P7T.

Smaller increases are also achieved for Bikel's parser, when trained on the small training set of the MFT. When Bikel's parser carries out its own POS tagging, it scores 0.3% better, and when unknown POS tags are supplied, it performs 2.51% better than its counterpart trained on the large training set of the P7T.

Table 7 also shows how scores using Bikel's parser jump, once again, when run in perfect tagging mode. Arun and Keller do not report results for running Bikel in perfect tagging mode.[18]

The variances in the increases of f-scores seem to be the direct results of the parsing mechanisms adopted by each of the parsers. BitPar is less flexible to inconsistent and error-ridden data, than Bikel's parser, which assumes independence relations among sister nodes, compensating for this with only a distance measurement.

The learning curves in Figure 7 present the changes in parser performance trained on increasingly large subsets of the MFT training set. For this experiment, we also train on the development set to obtain further information about possible increases in parser performance and its possible simple correlation to training set size.

Due to the small number of observations, any nonlinear growth curve fitting method would be parsimonious, we therefore applied linear regression analysis. Using four different combinations of power transformations, we found these learning curves to be approximately linear with a very strong positive relationship between transformed number and f-score. Table 8 shows the transforms, R^2, parameters, and parameter p-values (using the standard t-test). F-score extrapolation for a training set of size 18,548 (the size of the training set for experiments by Arun and Keller on the P7T) are given in Table 9. These predictions show an increase in f-score across the board.[19]

Parser and Mode	P7T f-score	MFT predicted f-score
BitPar (own POS tagging)	64.42	75.72
BitPar (perfect tagging)	67.42	81.08
Bikel (own POS tagging)	79.65	82.44
Bikel (unknown POS supplied)	80.50	83.99

Table 9: F-score and f-score prediction comparison for training set of size 18,548

The largest increase between MFT parsing scores and predicted parsing scores with a larger training set is for BitPar, whose predicted score is 11.3% higher when doing its own POS tagging, and 13.66% higher in perfect tagging mode. In fact, the performance gap between BitPar and Bikel's parser seems to be steadily closing as MFT training data sizes increase. These results suggest that lexicalisation for statistical parsing of French is perhaps not as crucial as was concluded by Arun and Keller (2005).[20]

7 Conclusion

We have presented the Modified French Treebank, a new French Treebank, derived from the P7T, which is cleaner, more coherent, has several transformed structures, and introduces new linguistic analyses. The positive effect of transformations on and cleaning up treebanks is well documented (for example, by Dickinson and Meurers (2005)). We investigated one important effect of a clean treebank on corpus linguistics. The MFT provides an

[18] Bikel's parser can be tricked into perfect tagging mode, by appending the part-of-speech to the end of each wordform.

[19] Significance tests are not applicable.

[20] Some authors (for example, Rehbein and van Genabith (2007); Kűbler (2005)) argue that parsing results for treebanks with different annotation schemes are not comparable. However, this conclusion remains unaffected by such argumentation.

Parser and Mode	transform	R^2	α	p-value (α)	β	p-value (β)
BitPar (own POS tagging)	$y = \alpha \cdot ln(x) + \beta$	0.9978	6.7206	1.53E-10	14.8734	8.11E-07
BitPar (perfect tagging)	$y = \frac{1}{\alpha \cdot ln(x) + \beta}$	0.9616	-0.0003	3.283E-06	0.0156	1.1472E-11
Bikel (own POS tagging)	$y = \frac{\alpha}{ln(x)} + \beta$	0.9943	-298.6927	4.03E-09	115.4334	2.42E-12
Bikel (unknown POS supplied)	$y = \frac{ln(x)}{\alpha + \beta \cdot ln(x)}$	0.977	0.01551	5.42E-07	0.0102	8.32E-12

Table 8: Linear regression on learning curve data from Figure 7

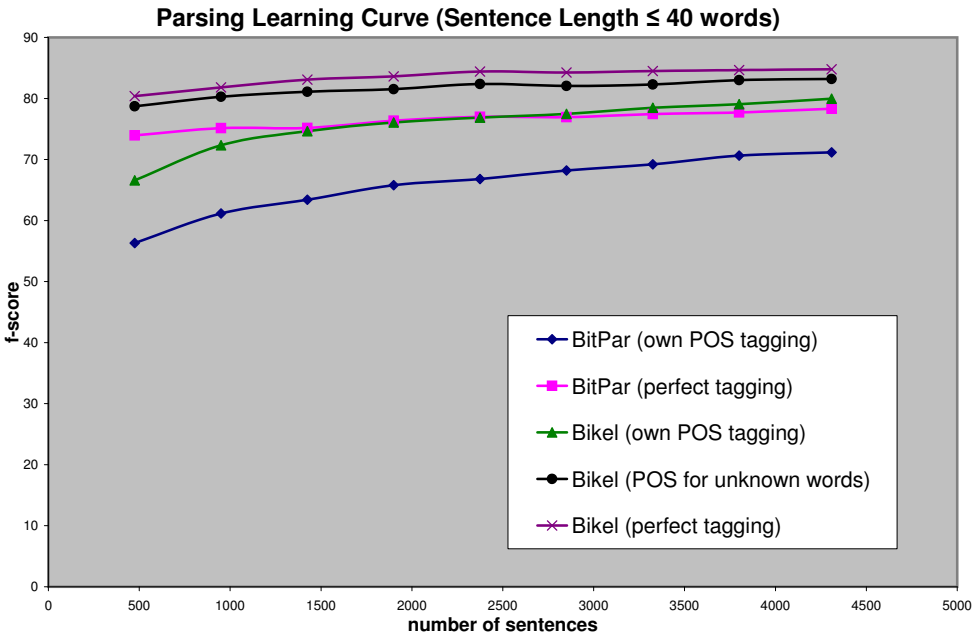


Figure 7: Learning Curve for the MFT.

extreme example of how quantity does not always make up for quality in statistical parsing. A probabilistic parser trained on clean and transformed data performs better than its counterpart trained on the original French treebank, which consists of five times the data. Moreover, we have shown how data which has a high error rate and that is not "parser-friendly" can lead to the potentially erroneous conclusions about the impact of lexicalisation on probabilistic parsing of French.

Acknowledgements

We gratefully acknowledge support from Science Foundation Ireland grant 04/IN/I527 for the research reported in this paper.

References

Anne Abeillé and Nicholas Barrier. 2004. Enriching a french treebank. In *LREC Conference Proceedings*. Lisbon.

Anne Abeillé, François Toussenel, and Martine Chéradame. 2004. Corpus le monde: Annotations en constituants. guide pour les correcteurs. Technical report, LLF and UFRL and Université Paris 7.

A. Arun and F. Keller. 2005. Lexicalisation in crosslinguistic probablisitic parsing: The case of french. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 306–313. Ann Arbor, MI.

A. Bies, M. Ferguson, K. Katz, and R. MacIntyre. 1995. Bracketing guidelines for treebank ii style penn treebank project. Technical report, University of Pennsylvania.

D. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the 2nd International Conference on Human Language Technology Research*. San Francisco.

Mary Dalrymple. 2001. *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press, San Diego.

Markus Dickinson and W. Detmar Meurers. 2003a. Detecting errors in part-of-speech annotation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-03)*, pages 107–114. Budapest, Hungary.

Markus Dickinson and W. Detmar Meurers. 2003b. Detecting inconsistencies in treebanks. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*, pages 45–56. Växjö, Sweden.

Markus Dickinson and W. Detmar Meurers. 2005. Prune diseased branches to get healthy trees! how to find erroneous local trees in a treebank and why it matters. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*. Barcelona, Spain.

Mark Johnson. 1998. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

S. Kűbler. 2005. How do treebank annotation schemes influence parsing results? or how not to compare apples and oranges. In *Proc. of RANLP*. Borovets, Bularia.

I. Rehbein and J. van Genabith. 2007. Treebank annotation schemes and parser evaluation for german. In *Proc. of the EMNLP-CoNLL 2007*. Prague, Czech Republic.

H. Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland.

Leonoor van der Beek. 2003. The dutch it-cleft constructions. In *Proceedings of the LFG03 Conference*.

Improving the complement/adjunct distinction in CCGbank

Matthew Honnibal and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{mhonn, james}@it.usyd.edu.au

Abstract

One of the challenges of adapting the Penn Treebank for a specific formalism is that the target annotation often requires information represented imperfectly or not at all in the original corpus. When this occurs, the information must either be guessed with heuristics, or annotated manually. Recently, a third option has become available, due to the release of resources that supplement the Penn Treebank, such as the predicate-argument annotation provided by PropBank.

In this paper, we discuss how such supplementary resources can be used to correct resources acquired from the Penn Treebank before they were available. We describe how we use the PropBank annotation to correct the complement and adjunct labels in CCGbank, producing cleaner, more consistent annotation. We also demonstrate that the new corpus can still be accurately reproduced by a statistical parser.

1 Introduction

The Penn Treebank's (PTB) annotation scheme was designed to make the corpus as widely useful as possible, but also to avoid drawing distinctions which the annotators found difficult to apply (Marcus et al., 1994). The corpus can be adapted to suit other formalisms, but problems will be encountered if the new annotation scheme requires information not represented in the original trees.

For instance, combinatory categorial grammar (CCG) requires binary branching trees, and distinguishes complements from adjuncts. The Penn Treebank does not describe the internal structure of noun phrases, and does not consistently apply complement and adjunct labels. Hockenmaier and Steedman (2005) created a CCG version of the PTB, but had to rely on heuristics in cases such as these, introducing noise into the resulting corpus.

Since the publication of the Penn Treebank, several projects have supplemented it by adding some of this missing information. Vadas and Curran (2007), for instance, have manually bracketed the noun phrases in the corpus whose structure was ambiguous. Another example is PropBank (Palmer et al., 2005), which adds predicate-argument annotation to the Penn Treebank. Recently, Shen et al. (2007) used PropBank to create a novel LTAG corpus from the Penn Treebank, LTAG-spinal.

In this paper, we show that such supplementary resources can be used to reform a corpus acquired from the Penn Treebank, correcting noise introduced by conversion heuristics. We use the semantic role labels in PropBank as the basis for the syntactic distinction between complements and adjuncts CCG requires. In order to do this, we first create a version of PropBank's stand-off annotation that refers to nodes in CCGbank, rather than nodes in the Penn Treebank. We then find arguments whose CCGbank label is inconsistent with their PropBank label, and apply changes to the CCG derivation.

When making these changes, we must avoid producing faulty derivations. Most CCG productions are licensed by a small set of general combinatory rules. To avoid creating productions unlicensed by these rules, we must propagate changes down the derivation tree. The rules for doing this, and for selecting the most appropriate new node label, must be as robust to noise in PropBank and CCGbank as possible.

To evaluate the resulting corpus, we check how often the changes have produced invalid derivations, and how consistently the reformed distinction between complements and adjuncts can be drawn by a parser. Validation is performed by

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 210–217,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

checking whether each production in the sentence occurred in the original corpus, or is licensed by the general combinatory rules. We find that only 22 sentences fail this validation process – a tiny proportion of the 10,254 sentences we have changed. We also find that the new complement and adjunct distinction does not dramatically impact the accuracy of a C&C parser (Clark and Curran, 2007) trained and evaluated on the modified corpus.

2 The Importance of Accurate Complement/Adjunct Labels

'Deep' grammars, such as LFG, HPSG, CCG and LTAG, attempt to generate semantic analyses, instead of skeletal parses. This typically involves producing a predicate-argument structure or logical form of the sentence. By contrast, 'shallow' parsers pay little attention to the interface between grammar and semantics, leaving the semantic extraction as a task for some downstream process.

The distinction between a complement — a generally obligatory, core argument of the predicate — and an adjunct — an optional argument that provides some auxiliary information from a restricted set of categories — is one of the tasks of this semantic analysis. Since a statistical parser learns its distinctions from the examples it is given, the accuracy of a 'deep' parser's analysis therefore depends to some extent on the accuracy of the complement and adjunct labels in its training corpus.

It is particularly useful for the corpus's complement and adjunct labels to be drawn from a semantic analysis that includes more information than the parser provides. PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998) are two examples of such analyses. If the parser is designed to replicate part of the annotation of one of these resources, it will then be easier to recover the rest of the information. For instance, if the argument structure of each verb in CCGbank is made to replicate the argument structure of each predicate in PropBank, the only missing information are the finer grained semantic role labels.

3 Combinatory Categorial Grammar

CCG (Steedman, 2000) is a linguistically motivated formalism that extends Categorial Grammar (Bar-Hillel, 1953) with a small set of operations, most of which were inspired by combinatory logic (Curry and Feys, 1958). Each word is assigned a category, and the categories are then combined according to a small set of rules to form the derivation. In the original Categorial Grammar, there are only two combinatory rules, forward application and backward application:

$$X/Y \; Y \;\Rightarrow\; X \quad (>) \tag{1}$$

$$Y \; X\backslash Y \;\Rightarrow\; X \quad (<) \tag{2}$$

where X and Y denote categories (either basic or complex). Figure 1 shows a derivation using these two rules. The category of the main verb, *join*, can be regarded as a function from two *NP* arguments to a sentence. Categorial Grammar is context-free, and so does not adequately describe long-range dependencies. The extra combinators in CCG increase the generative power of the grammar to mildly context sensitive (Steedman, 2000).

Because every combinatory rule operates on exactly two categories, derivations are generally binary trees; although there are a few unary productions too. Unlike a phrase-structure grammar, production rules are not arbitrary pairings of parent and child labels: two categories can usually only be combined to form one parent, so if only one node label is changed the derivation will almost always be rendered invalid.

4 CCGbank

Hockenmaier and Steedman (2005) adapted the Penn Treebank into a CCG corpus, CCGbank. The project was similar to other theory-specific treebanks acquired from more general resources, such as Chen and Vijay-Shanker (2000)'s TAG corpus. The CCGbank annotation consists of a CCG derivations and corresponding dependencies for 96% of the sentences in the Penn Treebank. The derivations are binary trees that record the categories, but not the combinatory rules invoked.

One of the main difficulties in the adaptation this was that a CCG analysis often encodes more linguistic information than is present in the original tree, as CCG relies on distinctions which the Penn Treebank annotators found too costly to reliably annotate. For instance, Hockenmaier employs head-finding heuristics much like those described by Magerman (1995) to binarise noun phrases, since the internal structure of noun-phrases is often ambiguous in the Penn Treebank description. Another source of difficulty were

structures where the original analysis differed substantially from the desired CCG analysis. There are many constructions which receive non-traditional analyses in CCG, as the properties of the formalism present different possibilities. Extraction and coordination constructions are prominent examples of this.

Constituent function labels are another case where the CCG formalism offers a different analysis from a phrase-structure grammar. In the Penn Treebank, function labels are encoded as extra information on certain nodes. This information is difficult to encode in a phrase-structure grammar, so it is generally not used by parsers trained on the original treebank, and the function labels were not rigorously checked and corrected when the corpus was created (Bies, 1995).

In CCG, however, the valency of a verb is represented in its category, and so the distinction between complements and adjuncts is represented in the derivation. Adjuncts are categories of the form $X\backslash X$ or X/X. That is, they are functions that take a category and return it unchanged, allowing unlimited other adjuncts to attach. Complements are usually atomic categories that are arguments of the verb. When CCGbank was created, complements and adjuncts were distinguished using the inconsistent function labels in the Penn Treebank, introducing noise into the corpus.

Several statistical parsers have been trained and evaluated on CCGbank. We use the C&C parser described by Clark and Curran (2007) to examine how the changes we have made have affected this use of the corpus.

5 PropBank

PropBank (Palmer et al., 2005) provides predicate-argument structure annotation for each sentence in the Penn Treebank. This annotation involves identifying each predicate in the sentence, and which constituents realise its arguments. The arguments then receive detailed labels that specify their semantic role, capturing generalisations that are complicated by surface syntactic constructions such as passivisation and ergativity.

The label scheme begins by describing each argument as either core or peripheral. Core arguments receive a numeric semantic role label, the interpretation of which is specific to each predicate. Peripheral arguments receive a label from a generic set. Peripheral arguments are adjuncts, and are defined by the fact that they are optional, cannot become the subject of the predicate, and can be assigned a label from a small set generic to all predicates, such as direction, purpose and time.

PropBank is a semantic annotation layer, so the target of the argument labels may not be a single constituent. Annotation is provided in a stand-off format, and labels may refer to multiple nodes, and often include references to traced constituents. However, syntactic distinctions are given priority over semantic differences when determining the frame set of a particular predicate, in order to avoid making distinctions that are too fine grained.

6 Aligning CCGbank and PropBank

In order to use the PropBank annotation for CCGbank, we first align the two resources. First, we align the sentences and tokens in CCGbank with the original PTB text, so that the few sentences missing from CCGbank do not cause the PropBank indices to refer to the wrong sentences. Token alignment is necessary too, because CCGbank sentences do not include the PTB null elements. CCGbank also omits quotation marks.

Once we have the correct sentences and token indices, we must select a set of CCG nodes that correspond to the PTB nodes that realise an argument. Since we cannot expect the CCG derivation to match the structure of the original parse, we search for the smallest set of CCG nodes that cover all and only the tokens covered by the PTB nodes PropBank refers to.

To do this, we first collect the tokens covered by the PTB nodes that realise the PropBank argument. For each word, we move up the derivation until we find a node whose parent includes at least one token not in the set we are looking for. We then add this node to the set of CCG nodes that will realise the argument. If two or more tokens in the set have a common ancestor that does not span any tokens not in the set, that ancestor will be referred to instead of the tokens it yields.

7 Changing CCGbank

This section describes the process we used to find inconsistencies between PropBank and CCGbank, and how changes were applied.

7.1 Changing node labels

In a phrase-structure grammar, production rules are arbitrary and unbounded: there is no underly-

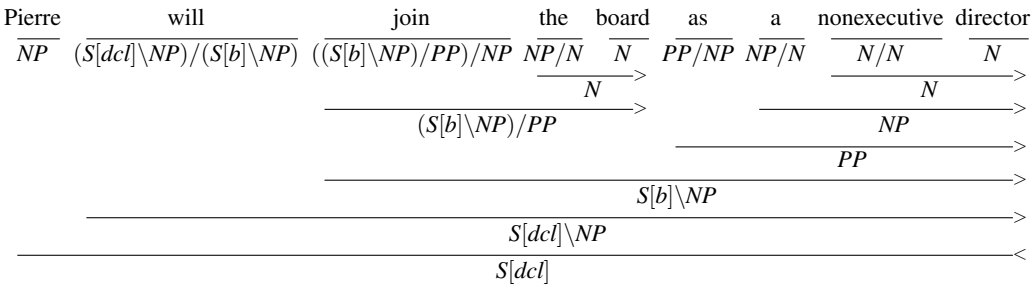


Figure 1: The original CCGbank derivation of the first sentence of the Penn Treebank.

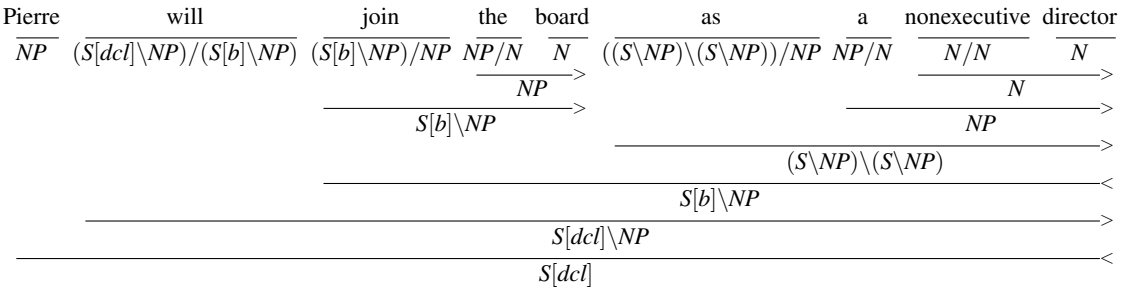


Figure 2: The altered derivation, with the *PP* argument changed to an adjunct.

ing logic behind why one symbol should be rewritten as the set of symbols that make up its children. As discussed in section 3, this is not true in CCG. We cannot simply replace one node label with another, or we will introduce an invalid production rule. When a node label is changed, the changes must be reflected in the surrounding tree.

The sibling of the node we are changing will always be the verb, as discussed in section 7.2. The new verb category will be produced by adding or deleting an argument, depending on whether we are changing an adjunct to a complement or vice versa. The rules for deciding the new sibling category are therefore given in sections 7.3 and 7.4.

We do this by finding instances of the old label in the subtree, and replacing them with the new label. We must replace the old category if it occurs as a whole label, a result, or an argument. Figure 3 shows a tree where the parent category is produced by using the forward application rule on the two children. This means that the parent category will be the result category of the left child, and the left child's argument category will be the right child's whole label.

Unary productions present a special case not covered by the rules discussed above. Unary productions are rules where the parent has only one child.

CCG does license unary type-raising rules, and Hockenmaier and Steedman (2005) discusses the

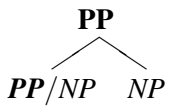


Figure 3: The parent category is the result category of the left child.

addition of several extra unary rules to simplify the category set. Other unary productions seem to be the result of noise. In the latter two cases, there is no obvious relationship between the parent category and the child category, as there would be with a binary production licensed by the grammar. This prevents us from using the rules discussed above to propagate the changes down the tree.

Figure 4 shows a tree which we would like to change to an argument, without introducing a new production rule NP → N. If we simply change the parent category and not the child, we will probably be introducing a new rule into the grammar. Since we wish to avoid this, we instead collapse the unary production, moving the grand-children up to the node whose label we have just changed, before propagating the changes to them. Figure 5 shows the tree after we have applied this process.

7.2 Finding arguments

This section discusses how we identify inconsistencies between the PropBank and CCG argument structure annotation. We iterate through

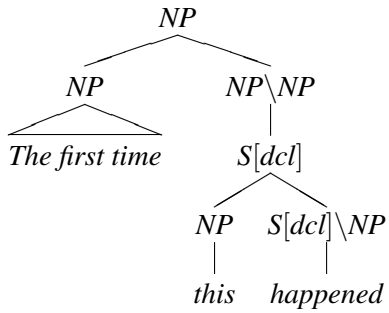


Figure 6: PropBank annotates 'The first time' as a peripheral argument of 'happened'. We do not change cases like this.

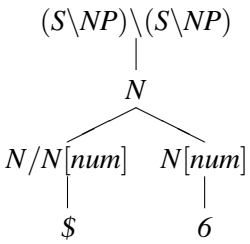


Figure 4: An adjunct with a unary production.

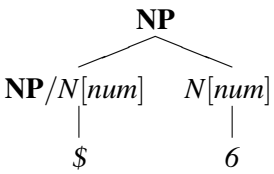


Figure 5: The tree relabelled as a complement, with the unary production collapsed and changes propagated through the subtree.

each PropBank argument, using several criteria to decide whether its CCG category ought to be changed.

The first criterion is that the PropBank argument should be realised by a single CCG constituent. This excludes 1.6% (6,271) of arguments from consideration. We are not interested in these arguments, because we are looking for cases where we can identify a simple inconsistency between the PropBank label and the CCG category. If there is not a single constituent, the two annotations disagree more fundamentally than our rules are designed to correct. Often, it would not even make sense to consider the PropBank argument as a syntactic constituent, such as the many cases where there is intervening text between the constituents that realise it.

Next, we check that the CCG category is consistent with the PropBank label. If the PropBank label is ARGM, the constituent should have a sentential adjunct category like $(S\backslash NP)\backslash(S\backslash NP)$, S/S, etc. Sentential adjunct categories are distinguished by having identical results and arguments, and having an innermost result category of S. If the argument has a numeric PropBank label, indicating that it is a subject or complement, its constituent should not have a CCG adjunct category unless it is the agent of a passive verb. 3.9% (15,686) of PropBank arguments have labels inconsistent with their CCG categories in this way.

The other criteria we consider are intended to guarantee that in the CCG derivation the argument attaches directly to the predicate, as either a CCG adjunct or a CCG argument. First, we check that the sibling of the constituent is a verb. The first criterion we use to do this is to check that its category's innermost result is S. Next, we check that the category is not an adjunct, by checking whether its argument is identical to its result. We also do not wish to change or introduce leftward arguments, so we add an extra constraint rejecting arguments which occur before the verb they attach to. This constraint is very rarely required, rejecting a further 10 arguments only.

6% (946) of the 15,686 arguments with inconsistent PropBank and CCGbank annotations fail these criteria. The most common case where this criterion is not met is when a relative clause is attached to a noun that PropBank considers a peripheral argument. Figure 6 shows a tree fragment that exemplifies this. 62.2% (583) of the 936 cases described above are like this example.

Once we have established that the constituent is attached to a verb, we check that the head of that verb node is the predicate the PropBank entry refers to. The case this corrects is rare: only 0.7% (98) of the 14,750 remaining inconsistencies exhibit this problem. Nevertheless, catching this case greatly reduces the number of invalid deriva-

tions our changes produce.

7.3 Changing complements to adjuncts

Changing a complement to an adjunct involves changing the constituent's label, the verb label, and updating the dependencies. The new constituent label is always $(S\backslash NP)\backslash(S\backslash NP)$, the standard non-fronted adjunct category. Our constraints ensure that the constituent occurs after the verb and that the verb category is not atomic.

As discussed above, the verb node is the sibling of the constituent. The new category of this constituent will be its current result, as we are simply changing the argument to an adjunct. This also ensures that the parent category will remain unchanged.

These changes alter which CCG rules are invoked in the derivation. In the simplest case, the verb and the constituent no longer combine by forward application of the verb category, but by backward application of the new adjunct. Figure 2 shows an example of this.

Occasionally, however, the new derivation may require backward composition, as the verb could still looking for rightward arguments. Table 1 lists the instances of the various combinatory rules in CCGbank derivations before and after our changes, showing that despite our initial concern, the new corpus actually contains slightly fewer productions validated by composition rules.

Having changed the derivation, we must update the dependencies it produces. We must delete one local dependency, and introduce one local dependency, as instead of the verb governing the argument, the argument is now an adjunct – a functor category that governs the verb.

We change 1,543 complements to adjuncts.

7.4 Changing adjuncts to complements

Changing an adjunct to a complement involves selecting the new label of the complement, adding it as an argument to the verb category, and updating the dependencies. Selecting the appropriate label for the complement can be tricky, as CCG categories reflect the constituent's function, not its form. An *NP* functioning as an adjunct and a *PP* functioning as an adjunct will therefore both have the same category. The first heuristic we use deals with adjuncts formed by unary productions. In these cases, the adjunct node is essentially replaced by its child, as we simply need to undo the unary rule. Otherwise, we try to decide whether the complement is to be labelled *NP* or *PP* by checking whether its head is a noun, based on its part of speech tag.

Rule	Before	After
F. App.	674,487	686,209
B. App.	227,552	217,621
F. Comp.	4,834	4,835
Gen. F. Comp	21	21
F. X. Comp	51	51
Gen. F. X. Comp	1,314	1,226
B. Comp.	1,601	1,520
Gen. B. Comp	237	238
B. X. Comp.	13,582	11,841
Gen. B. X. Comp.	1,560	1,529
Total valid	1,094,249	1,094,303
Total invalid	5,243	5,189

Table 1: The frequency with which CCG rules are invoked to validate production rules before and after our changes. F.=Forwards, B.=Backwards, App.=Application, Comp.=Composition, X.=Crossing, Gen.=Generalised.

Once we have the category of the complement, we can add it as an argument to the verb. The new verb category will be of the form X/Y, where X is the old verb category, and Y is the complement category.

Having changed the derivation, we must update the dependencies it produces. We must delete one local dependency, and introduce one local dependency, as instead of the complement governing the verb as an adjunct, the verb now governs the complement as an argument.

We change 13,256 adjuncts to complements. Many of these are arguments of verbs of price motion.

8 Validating the changes

Once we have changed the corpus, we wish to know whether our alterations have had an adverse effect. We approach this issue by asking how often our changes have produced derivations not licensed by the original grammar. Validation proceeds in two steps. First, we check whether each production in the new corpus occurred in the original. If a production did not occur in the original corpus, we check whether it is licensed by the general combinatory rules.

Novel productions are sometimes introduced when changing adjuncts to complements intro-

duces verb categories with novel argument structures. Other novel productions seem to be the result of the unpredictable interaction of our algorithm with noise in CCGbank.

In order to further examine these novel production rules, we check whether they are licensed by the general combinatory rules of CCG. If the derivation passes this test, the changes are preserved; if not the changes are reverted. Very few sentences fail this validation process: only 22 out of 10,254 changed.

It is also interesting to apply the two validation processes in the opposite order. This also examines how many derivations in the original corpus are licensed by the CCG combinatory rules. We considered all unary productions valid for the purpose of this evaluation. We also added a few extra rules that deal with punctuation. These rules effectively treat punctuation marks as having schematic categories like X/X and $X\backslash X$, allowing them to be 'absorbed' into any category to their left or right.

Table 1 describes how often the various rules were invoked to license productions in the corpus before and after changes. It also shows how many productions could not be validated by the rules alone. The table shows that despite our concerns, our changes have not actually introduced more composition productions into the corpus.

9 Parsing results

Table 2 presents the results from evaluating the C&C parser on section 23 of CCGbank, before and after changes. The results show that the parser performs slightly worse on the new corpus. There are several possible reasons for this.

One possibility is that the increase in complements in the new corpus simply makes the task harder. Because verb categories reflect the predicate's argument structure, increasing the number of non-NP complements may make the supertagger's task of selecting the correct verb category slightly more difficult. On the other hand, the small difference in performance may simply be due to a more even balance between complements and adjuncts in the new corpus, since prepositional phrases are usually adjuncts.

Another possible explanation for reduced performance is that the parameters used in the supertagger and parser, e.g. levels of ambiguity used in supertagging, have been optimised for the original CCGbank, and we are yet to repeat this process for the new corpus.

10 Conclusion

Treebanks are expensive projects, and limited resources are available for their construction. This typically forces compromises to be made in the construction of a treebank, both in the information its annotation scheme is designed to represent, and how rigorously that annotation scheme is enforced.

One of the most common compromises made is to acquire a theory specific corpus from a more general one. This is a very effective strategy, but one that generally introduces some level of noise into the resulting corpus. One of the principle sources of such noise is distinctions made in the new annotation scheme that are only partially supported by the original corpus.

We have shown how new information can be incorporated into a corpus automatically acquired from the Penn Treebank, in order to mitigate this problem. As supplementary resources are released for the Penn Treebank, corpora calculated from it can be updated, thereby benefiting from these steady improvements.

Improving the training data with respect to this distinction may also improve parsing results. Clark and Curran (2007) report that the distinction between complements, adjuncts and NP qualifiers is one of the main sources of error. To some extent, this is unsurprising, as PP attachment is one of the most difficult distinctions to draw. However, the inconsistent complement and adjunct labels that CCGbank inherits from the Penn Treebank may well be a problem. Although our current results show little change in parsing performance, we believe future experimentation will see the benefit of making CCGbank more consistent.

We have demonstrated how the practical issues involved in reforming a problematic linguistic distinction can be overcome. We have described how the three resources required — CCGbank, the Penn Treebank, and PropBank — can be aligned; and how the PropBank labels can be used to correct the CCGbank derivations. The reformed complement and adjunct labels, which sometimes involve more subtle semantic distinctions drawn by the PropBank annotators, can be replicated by a parser almost as accurately as the original annotation, without tuning the parser to better handle the changes in the corpus.

Model	LP	LR	LF	SENT	UP	UR	UF	CAT	cov
Original	85.75	84.88	85.31	32.28	92.53	91.59	92.05	93.19	99.27
Modified	84.31	82.91	83.61	29.24	92.53	90.99	91.75	93.00	99.22

Table 2: Results on Section 00 with both models using normal-form constraints

11 Acknowledgements

We would like to thanks the anonymous reviewers for their helpful comments. James Curran was funded under ARC Discovery grants DP0453131 and DP0665973.

References

Colin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 86–90. Montreal, Canada.

Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

A. Bies. 1995. Bracketing guidelines for Treebank II style. Penn Treebank Project.

J. Chen and K. Vijay-Shanker. 2000. Automated extraction of tags from the penn treebank. In *Proceedings of the 6th International Workshop on Parsing Technologies*. Trento, Italy.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*. (to appear).

Haskell B. Curry and Robert Feys. 1958. *Combinatory Logic: Vol. I.* Amsterdam, North Holland.

Julia Hockenmaier and Mark Steedman. 2005. Ccgbank manual. Technical Report MS-CIS-05-09, University of Pennsylvania.

David Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Meeting of the ACL*, pages 276–283. Cambridge, MA.

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.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Libin Shen, Lucas Champollion, and Aravind K. Joshi. 2007. LTAG-spinal and the treebank. *Language Resources and Evaluation*. to appear.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

David Vadas and James R. Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 240–247. Prague, Czech Republic.

Evaluating Feature Types for Encoding Clinical Notes

Jon Patrick
School of IT
University of Sydney
Australia

Yitao Zhang
School of IT
University of Sydney
Australia

Yefeng Wang
School of IT
University of Sydney
Australia

{jonpat, yitao, ywang1}@it.usyd.edu.au

Abstract

This paper proposes a machine learning approach to the task of assigning the international standard on classification of diseases ICD-9-CM codes to clinical records. By treating the task as a text categorisation problem, a classification system was built which explores a variety of features including negation, different strategies of measuring gloss overlaps between the content of clinical records and ICD-9-CM code descriptions together with expansion of the glosses from the ICD-9-CM hierarchy. The best classifier achieved an overall F_1 value of 88.2 on a data set of 978 free text clinical records, and was better than the performance of two out of three human annotators.

1 Introduction

Despite the rapid progress on text categorisation in the newswire domain, assigning meaningful labels to clinical notes has only recently emerged as a topic for computational linguists although health informatics researchers have been working on the problem for over 10 years. This paper describes constructing classifiers for the Computational Medicine Center's 2007 Medical Natural Language Processing Challenge which aims to assign ICD-9-CM codes to free text radiology reports. In this work we distinguish between "medical" and "clinical" texts, where the former refers to carefully written prose about medical knowledge such as published books or papers, the latter is prose often ungrammatical written by a clinician as part of the record for point of care attendance of a patient. As such it includes things like bedside notes, various types of medical measurement reports, (e.g. radiology, pathology, sonography, etc.) and discharge summaries.

A variety of machine learning algorithms, such as decision trees, maximum entropy modeling, and support vector machines, have already been applied to text categorisation tasks (Apté et al., 1994; Nigam et al., 1999; Joachims, 1998). Most of these systems use bag-of-words features which generally neglect the internal structure of texts. Other features like representations based on phrases (Dumais et al., 1998; Furnkranz et al., 2000), and named entities (Kumaran and Allan, 2004) have also been proposed. However, these techniques are only moderately effective and the performance of text classification in newswire domain has reached a plateau.

Although bag-of-words features work well in many general text categorisation tasks, their effectiveness is limited in the clinical domain because of the different characteristics between clinical text and newswire text, for example, the negation statements in medical reports. The presence of a concept in texts does not necessarily mean that the finding or disease is definite. The concept may refer to a finding that is absent or a procedure not chosen to be used. Many of the findings in clinical reports are actually statements of negation of exclusion by clinicians, or a disease can be ruled out or inconsistent with a suspected disease. The negation problem has been addressed by previous works in information retrieval and decision support systems (Chapman et al., 2001; Elkin et al., 2006).

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 218–225,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

The varying terminology also cause problems for understanding medical texts. The same medical concept can be represented with different surface terms. For example, the concept "cancer" has more than 20 synonyms, one of them is "malignant tumor" which is lexically different from "cancer". Since a bag-of-words modeling approach only uses pieces of information that are explicitly mentioned in texts, it will neglect the existence of "cancer" as an important feature if only "malignant tumor" is found in the training data set.

This paper addresses the difficulties of medical text categorisation tasks by incorporating medical negations, term variations, and clues from hierarchy of medical ontologies as additional features. Section 2 introduces the details about the task and the data set used in our experiments. Section 3 describes various feature types. Section 4 and Section 5 explains the experiment settings and result respectively. Finally, the paper discusses the experimental results in detail in Section 6.

2 The task of assigning ICD-9-CM codes

2.1 Data set

The corpus used in this study is a collection of radiology reports from the Cincinnati Children's Hospital Medical Center, Department of Radiology. (Computational Medicine Center, 2007) The data set is divided into a training set and a test set. The training set consists of 978 records and the test set consists of 976 records. However, since the gold-standard codes of the test set have not been released yet, this paper considers the training set as the only source for both training and testing of our system.

Each clinical record has two parts: "Clinical History" which is provided by an ordering physician before a radiological procedure, and "Impression" which is reported by a radiologist after the procedure. Three different companies were invited to assign ICD-9-CM codes for each record. A clinical record can be assigned with more than one ICD-9-CM codes at a time. The majority code with at least two votes from the three annotators was considered as the gold-standard code for the record. All clinical records in the corpus were anonymized.

There are altogether 45 ICD-9-CM codes found in the corpus. However, the distribution of codes in the corpus is imbalanced. There are four codes which occur more than 100 times in the training set and there are also 10 codes with a frequency of only one. As a consequence, we decided to train our machine learning system only on ICD-9-CM codes which occur at least twice in the training set.

2.2 The machine learning approach

The task of assigning ICD-9-CM codes to clinical records was considered as a multi-label text categorisation problem. For each code found in the corpus, we created a separate classifier which makes binary "Yes" or "No" decisions for the target code of a clinical record. Maximum Entropy Modeling (MaxEnt) (Berger et al., 1996) and Support Vector Machine (SVM) (Vapnik, 1995) were used to build the classifiers in our solution.

3 Features

Due to the clinical nature of the texts, a variety of features were developed to represent what we believed were the important determiners of the ICD codes. Table 2 and Table 3 lists all the features used in the experiments.

3.1 Bag-of-words feature

Bag-of-words (BOW) features include only unigrams and bigrams in the text. Each BOW feature was given a "Clinical_history" or "Impression" prefix to represent the source it originated from. All words were normalized into their lower-case orthography. Stemmed-words were used as features in some experiments.

3.2 Negation feature

It is critical for the classification system to distinguish between medical concepts that are mentioned as being present and those that are mentioned as being absent, often by means of negation. For instance, if a classifier sees "no pneumonia" in a clinical record, it should not give the record the '486 Pneumonia, organism unspecified' code. However, using only bag-of-words modeling will assign this record with a "pneumonia" unigram feature which adds extra weight for encoding the record with "486". Although introducing a bigram "no pneumonia" might capture the possible negation, it is hard for the system to capture longer distance negations like "no ev-

no, without, no further,
absence, no sign, consistent with,
rule out, doubt, no evidence of,
versus, without indication of,
no history of, evaluate for, concern for

Table 1: Example of Negation Phrases

idence of pneumonia" by simply adding higher order ngram features.

Negation features were used in the classification system to capture the terms that are negated or uncertain. We created a negation-finding system which uses an algorithm similar to (Chapman et al., 2001) to identify the negation phrase and the scope of negations. Table 1 shows the list of explicit negation markers and possible markers which are manually created for the task.

When the negation-finder sees the following sentence:

> "Chest x-ray is normal, no evidence of pneumonia or viral disease."

it will identify "no evidence of" as a negation phrase, and "pneumonia" and "viral disease" as negated terms.

3.3 Gloss matching feature

The ICD-9-CM provides detailed text definition for each code. This section explores different strategies for measuring gloss matchings between the content of a clinical record and the definition of an ICD-9-CM code.

3.3.1 Target code, overlap code, and competitor code

According to the guidelines for encoding notes with ICD-9-CM codes (Computational Medicine Center, 2007), whenever there is a definite diagnosis found in a clinical record, only the diagnosis code should be assigned rather than any other symptom and sign codes that might co-occur in the text. For example, if a clinical record has already been given a definite diagnosis code of '486 (Pneumonia)', even if there are references to cough or fever, it should not be given any symptom and sign code like '786.2 (Cough)' or '780.6 (Fever)'. However, symptom and sign codes should be coded if the diagnosis in text is not definite. For instance, if pneumonia found in texts is only 'possible' or has been 'ruled out', then it should not be given a pneumonia code and all other symptoms and signs like cough and fever found in texts should be coded.

To capture this relationship between ICD-9-CM coding, this section introduces the notions of target code, overlap code, and competitor code.

Target Code: a target code is an ICD-9-CM code to which a binary classification system tries to assign a clinical record.

Overlap Code: an overlap code is the code which frequently co-exists with the target code in the corpus.

Competitor Code: a competitor code will act to block the co-existence of the target code.

For instance, if a system is currently making decisions on assigning '786.2 (Cough)' to a clinical record, then the current target code is 786.2. Its overlap codes include '780.6 (Fever)' because they tend to occur together in the corpus, while '486 (Pneumonia)' becomes a competitor code of the target code because if the clinical record has been given a definite diagnosis, it should not be coded with any symptom and sign code.

3.3.2 String match

There are two types of gloss matching features used in the experiments.

Binary Match: a binary match feature will be turned on if a piece of text matches the text definition of a code. We further distinguish between binary matches of two types of features: exact matches (f10, f20, f30 in Table 3), and partial matches (f11). Stemmed words in the source text were also compared to code definitions and f12, f13 in Table 3 are binary stemmed-word matching features.

Matched ngrams: matched ngrams were also used as another type of word overlap features. This type of feature emits all of the longest matched ngrams with code definitions, together with the local context of the matched ngrams. The intuition of considering the context of the matched ngrams is that it can automatically capture the possible negation terms in the context.

3.3.3 Synonym and acronym expansion

It is crucial for a system to identify synonyms and acronyms in texts which represent the same concept in different words. For example, 'febrile' and 'fever' are synonyms of code '780.6 Fever', and 'UTI' is the acronym of 'Urinary tract infection' which is important for identifying code '599.0 Urinary tract infection, site not specified'.

A synonym and acronym list from UMLS Specialist Lexicon and UMLS Metathesaurus (Humphreys and Lindberg, 1993) was prepared. For each word found in the corpus, the UMLS Specialist Lexicon was studied for its possible synonyms and acronyms. Then if the word also matched to a concept in the UMLS Metathesaurus, a list of all the different string representations of the concept was extracted as synonyms of the word. Finally, all the synonym and acronym pairs were filterer out which could not be matched with any ICD-9-CM code definitions.

3.3.4 ICD-9-CM hierarchy expansion

The ICD-9-CM hierarchy also provides important information about the classification of each code. For instance, codes ranging from 780 to 799 are all symptom and sign codes, and health history codes are all grouped under 'V' category like 'V13.02 Urinary (tract) infection'. The parent node of a target code in the ICD-9-CM provides a broader concept which can be considered as a hypernym. Therefore, we explored features like matching between text and the parent code of a target code, and the type of a target code like symptom and sign codes, health history codes, and disease codes which can be known from the root of the target code in the ICD-9-CM hierarchy.

4 Experiment

This section discusses the methods of our experiments on assigning ICD-9-CM codes to clinical records.

4.1 Preprocessing

The dTagger (Divita et al., 2006) was used to identify sentence boundaries and part-of-speech (POS) tags of each token. The Genia Tagger (Tsuruoka et al., 2005) was also applied for POS tagging and to identify chunks in the sentence which is crucial for finding possible negations in texts.

Bag-of-words (BOW) features	
f1	Unigrams
f2	Bigrams
Negation features	
f3	Negated words
f4	Negation words
Other	
f5	the target code is a sign/symptom code
f6	the target code is a history code

Table 2: Basic Features for the Classification Task

4.2 Classifier

Both MaxEnt and SVM classifiers were used in the experiments. The MaxEnt classifier used was developed by Zhang Le.[1]. For the SVM classifier was the SVM-light package (Joachims, 1999) with two different kernels: a linear kernel (SVM t=0) and a polynomial kernel (SVM t=1). The experimental results show that in most cases SVM t=0 outperforms both the MaxEnt and SVM t=1 classifier. Only the result of the best classifier for each individual ICD-9-CM code is reported.

4.3 Evaluation

The main evaluation metric is the microaveraged F_1 value which is the harmonic mean of precision and recall. In the experiments, the training data set of 978 records was randomly divided into ten folds. Ten-fold cross-validation was performed and all the true positives, false positives, and false negatives summed for each ICD-9-CM code in each testing fold to give the total TP, FP, and FN values for the whole data set.

5 Result

5.1 Baseline

Four baseline systems were created for the task.

Baseline 1: uniformly assigns the most frequent class '786.2 (Cough)' to all clinical records.

Baseline 2: uses partial overlap information between the text and the code definition. If a clinical

[1] http://homepages.inf.ed.ac.uk/s0450736/

Gloss matching features for the target code	
f10	Exact match
f11	Partial match
f12	Stemmed exact match
f13	Stemmed partial match
f14	UMLS synonyms match
f15	UMLS acronyms match
f16	Matched ngrams
f17	Previous context of the matched ngrams
f18	Next context of the matched ngrams
Gloss matching features for overlap codes	
f20	Exact match for overlap codes
f21	Partial match for overlap codes
f22	UMLS synonyms match for overlap codes
f23	UMLS acronyms match for overlap codes
f24	Matched ngrams of overlap codes
f25	Previous context of the matched ngrams
f26	Next context of the matched ngrams
Gloss matching features for competitor codes	
f30	Exact match for competitor codes
f31	Partial match for competitor codes
f32	UMLS synonyms match for competitor codes
f33	UMLS acronyms match for competitor codes
f34	Matched ngrams of competitor codes
f35	Previous context of the matched ngrams
f36	Next context of the matched ngrams
Gloss matching features for parent codes	
f40	Matched ngrams of parent codes
f41	Previous context of the matched ngrams
f42	Next context of the matched ngrams

Table 3: Gloss Matching Features for the Classification Task

record partially matches with a code definition after removing stop-words, it is assigned with the code. A threshold value of 0.1 was used as the lower-bound for matched words between the text and the code definition, normalized by the number of words of the code definition.

Baseline 3: is a single multi-classifier system which targets code categories rather than only single codes. A code category is any combination of single ICD-9-CM codes which constitute a disjointed set of labels for each clinical record in the corpus. For example, a clinical record assigned with two codes: 786.2 and 780.6, then it will automatically create a code category [786.2, 780.6]. We create a single MaxEnt multi-classifier for all code categories, and then produced the classification result for each individual code.

Baseline 4: creates a binary classifier for each ICD-9-CM code in the corpus and uses only BOW features. Unigrams and bigrams were used in the experiments. Results in Table 4 show that adding bigrams gives better performance. Both SVM and MaxEnt classifiers were used in the experiment but only the result of the best classifier for each ICD-9-CM code is reported.

5.2 Negation

Two different strategies for constructing negation features returned by the negation-finder were explored.

Add negations: The first strategy was to add separate features of both negation words and negated words. For instance, 'rule out pneumonia' will generate two features: 'Negation-words-rule out' and 'Negated-words-pneumonia'. The result of S1 in Table 4 shows this only introduced minor improvement on precision value.

Gloss matching on negated text: The second approach was to improve the quality of gloss matching features by removing the negated text from source texts. The result of S3 in Table 4 shows this approach has outperformed S1 by 1.2% in terms of F_1 value.

5.3 Gloss matching

The performance of different approaches of measuring overlaps between clinical records and ICD-9-CM code definitions were also studied.

Synonym and acronym expansion: Adding synonyms and acronyms on top of the BOW and text-overlap features improved the precision value by 0.7% (S4 in Table 4), while the recall value remained unchanged. This observation however contradicts our intuition that introducing more synonyms and acronyms should help reduce false negative instances and thus improving the recall rate. This result needs to be further explored.

Overlap and competitor codes: By adding matching information on overlap codes and competitor codes, S5 and S6, S7 in Table 4 show improvement of precision, and a slightly increase of recall by 0.2% over S4 which only performs matching of the target code.

Using the ICD-9-CM hierarchy: S7 in Table 4 shows the impact of using the hierarchical information provided by the ICD-9-CM hierarchy. Both the precision and recall rate drops significantly by introducing the matching information of the parent node of the target code. This occurs because most codes like '786.2 Cough' and '780.6 Fever' only need the correct matching of the target code definitions. Providing clues of matching between texts and the parent code definition will therefore only introduce more noise into the machine learning system. However, in cases like 'V13.09 Other', it might be crucial for the system to match its parent code 'V13.0 Disorders of urinary system' with texts in order to correctly assign the target code.

5.4 Feature engineering

In experiments with a uniform set of feature types for all ICD-9-CM codes, we noticed that different codes tend to have a preference for different combinations of feature types. For instance, code '786.2 Cough' has only a single word as its definition term, while the coding of '593.70 Unspecified or without reflux nephropathy' requires understanding its parent node '593.7 Vesicoureteral reflux', or maybe even '593 Other disorders of kidney and ureter'.

Therefore, different combinations of feature types for each individual code were used. The intuition is to explore different combination of feature types quickly instead of doing further feature selection procedures. The system trained on the best combination of feature types are reported as the final results for the target code. If an ICD-9-CM code has more than one best combination of feature types, we consider all classifiers trained on these best combinations and assign the code to a clinical record only if at least half of the classifiers have given the code to the record. The result of S8 in Table 4 shows that by using only the best combination of feature types significantly increases the recall rate from 80.6 to 86.0, while the precision rate has a minor improvement to 89.7.

5.5 Low frequency codes modeling

A rule-based system was also used to model low frequency ICD-9-CM codes which have only one occurrence in the corpus, or have achieved F_1 value of 0.0 by machine learning. The system assigns a low frequent code to a clinical record if the content of the record matches the words of the code definition. Moreover, if the system reports more than five instances of a particular low frequent code, all predictions of that code will be neglected. S9 in Table 4 shows that adding low frequency code modeling results on top of S8 gives 0.9% improvement on recall rate while precision remains almost unchanged.

6 Discussion

6.1 Comparison of annotator and machine learning system performances

In order to understand more about the ICD-9-CM code assignment task, this section evaluates the performance of the three annotators. Table 5 compares the performance of each annotator to the gold-standard codes. The item "N" in Table 5 stands for the total number of ICD-9-CM codes which an annotator has assigned to the whole corpus.

The result in Table 5 shows that our best reported machine learning system S9 has achieved very near human performance. This is probably due to different companies tend to have different strategies for assigning ICD-9-CM codes. Company1 and company2 tend to assign more ICD-9-CM codes, while company3 prefers a more conservative coding strategy. Also, these institutions seem to have different guidelines for assigning some specific codes. For instance, the disease code "599.0 Urinary tract infection, site not specified" and its corresponding health history code "V13.02 Urinary (tract) infection" are quite ambiguous codes which share many common

Name	Description	P	R	F_1
Baseline Systems				
B1	Majority code	27.2	21.8	24.2
B2	Simple gloss matching	24.8	82.6	38.1
B3	Disjointed multiclass labeling	61.9	56.1	58.9
B41	Unigram	83.9	78.4	81.1
B42	Unigram + Bigram	88.2	78.2	82.9
Systems with uniform feature types				
S1	B42 + negation	88.5	78.2	83.0
S2	B42 + gloss matching	87.7	80.0	83.7
S3	B42 + gloss matching on negated text	88.8	80.1	84.2
S4	S3 + Synonym, Acronym expansion	89.5	80.1	84.5
S5	S4 + competitor codes	89.0	80.3	84.4
S6	S5 + overlap codes	89.2	80.6	84.7
S7	s6 + father code	89.1	80.2	84.4
Systems with best individual feature types				
S8	best	89.7	86.0	87.8
S9	best + low-freq	89.7	86.9	88.2

Table 4: Experiment results for all ICD-9-CM codes

Name	P	R	F_1	N
company1	78.3	89.8	83.7	1397
company2	82.6	95.2	88.5	1404
company3	90.4	75.0	82.0	1011
system S9	89.7	86.9	88.2	1180

Table 5: Performances of Annotators

bag-of-words and gloss matching features. Company1 prefers to assign records on urinary tract infections with "V13.02" while other two companies prefer to code "599.0" in most cases. It looks like that they have adopted different guidelines of coding 599.0 and V13.02 within their institutions.

6.2 Unlabeled instance

There are 69 clinical records not given any ICD-9-CM code by our system. The following record (record ID = 99582148) gives an example of the unlabeled instances:

> CLINICAL HISTORY: Persistent cough, no fever.
> IMPRESSION: Retained secretions vs atelectasis in the right lower lobe. No infiltrates to support pneumonia.

The gold-standard label for this record is "786.2 Cough". The system did not assign the correct code probably because the text contains matches to code "518.0 Pulmonary collapse" which has a synonym "atelectasis". Our system did not break the tie between of 786.2 and 518, and decided to leave both codes unassigned.

We explored two different approaches for handling the unlabeled clinical records: assign an unlabeled instance with the code which has the highest probability, or give it any code which has the exact string-level match to the content. However, both strategies did not work well in that they have introduced far more false positives than number of decrease in false negatives. The system suffered a significant drop in precision without gaining a compensatory improvement of recall value. Therefore, we decided not to try to assign any code to these unlabeled clinical records.

6.3 Sophisticated paraphrase and entailment

We also noticed some examples which require more sophisticated modeling of paraphrasing and entailment in the medical domain. For example, the correct classification of "788.30 Urinary incontinence, unspecified" requires the system to understand that "voiding dysfunction" is a paraphrase of "urinary incontinence", and "anuresis" which is the inability to urinate is logically entailed by 788.30 as well.

7 Conclusion

This paper presents an approach to the problem of assigning ICD-9-CM codes to free text medical records. We created a classification system which consists of multiple machine-learned classifiers on high-frequency codes, and a rule-based modeling module of low-frequency codes. By incorporating negations and a variety of gloss matching features, we successfully outperformed the baseline with only bag-of-words features by 5.3% on F_1 value. The best reported score is also considered as comparable to the performance of the best human annotator. We also consider the way our system selected the best combination of feature types for each individual ICD-9-CM code has a major contribution to the classification task of clinical records.

References

C. Apté, F. Damerau, and S.M. Weiss. 1994. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, 12(3):233–251.

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.

W.W. Chapman, W. Bridewell, P. Hanbury, G.F. Cooper, and B.G. Buchanan. 2001. A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries. *Journal of Biomedical Informatics*, 34(5):301–310.

Computational Medicine Center. 2007. 2007 Medical Natural Language Processing Challenge. http://www.computationalmedicine.org/challenge/.

Guy Divita, Allen C. Browne, and Russell Loane. 2006. dTagger: A POS Tagger. In *AMIA 2006 Symposium Proceedings*, pages 200–203.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155.

P.L. Elkin, S.H. Brown, B.A. Bauer, C.S. Husser, W. Carruth, L.R. Bergstrom, and D.L. Wahner-Roedler. 2006. A controlled trial of automated classification of negation from clinical notes. *feedback*.

J. Furnkranz, T. Mitchell, and E. Riloff. 2000. A case study in using linguistic phrases for text categorization on the WWW. *Learning for Text Categorization: Proceedings of the 2000 AAAI/ICML Workshop*, pages 98–05.

B.L. Humphreys and D.A.B. Lindberg. 1993. The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association*, 81(2):170.

T. Joachims. 1998. Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142.

Thorsten Joachims. 1999. Making Large-scale SVM Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods-Support Vector Learning*.

G. Kumaran and J. Allan. 2004. Text classification and named entities for new event detection. *Proceedings of the 27th annual international conference on Research and developement in information retrieval*, pages 297–304.

K. Nigam, J. Lafferty, and A. McCallum. 1999. Using maximum entropy for text classification. *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a Robust Part-of-Speech Tagger for Biomedical Text. In *Advances in Informatics - 10th Panhellenic Conference on Informatics*, pages 382–392.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Global Revision in Summarisation: Generating Novel Sentences with Prim's Algorithm

Stephen Wan[†‡] **Robert Dale**[†] **Mark Dras**[†]
[†]Centre for Language Technology
Div of Information Communication Sciences
Macquarie University
Sydney, NSW 2113
swan,madras,rdale@ics.mq.edu.au

Cécile Paris[‡]
[‡]Information and Communication Technologies
CSIRO
Sydney, Australia
Cecile.Paris@csiro.au

Abstract

In abstract-like summarisation, extracted sentences containing key content are often revised to improve the coherence of the overall summary. In this work, we consider the task of Global Revision, in which a key sentence is revised and supplemented with additional content from the original document. Specifically, this task comprises two subtasks: selecting content; and grammatically ordering content, the focus of this paper. Using statistical dependency models, we search for a Maximal Spanning (Dependency) Tree that structures recycled words and phrases to form a novel sentence. Combining a modified version of Prim's algorithm with a four-gram language model, we evaluated our system on a sentence regeneration task obtaining Bleu scores of .30, a statistically significant improvement above the baseline.

1 Introduction

Faced with a seemingly endless supply of online textual information, it is often difficult to wade through a sea of documents returned by a search engine to identify those that are truly relevant to one's information need. To address this issue, automatic text summarisation technology strives to help provide shortened surrogates of documents that may be read quickly in order to make such judgements.

The current state-of-the-art approaches to summarisation simply return a list of extracted sentences. Whilst this is useful for identifying the gist of the document, this type of summary suffers from certain problems that make it difficult to glean more specific information about the original document. Typically, issues arise stemming from the fact that sentences are extracted in isolation and out of their original contexts. For example, an anaphoric reference in the extracted sentence may be missing the necessary antecedent to allow proper interpretation.

To combat such limitations, approaches to abstract-like summarisation often attempt to handle various additional linguistic phenomena in order to assist the reader make sense of the extracted pieces of information. Such methods range from accounting for discourse structure to maintain coherence (Marcu, 2000), to revising extracted text for fluency (for example, see Jing and McKeown (1999)). In this paper, we examine novel methods for accomplishing the task of revision as a text-to-text transformation.

Specifically, our overall goal is to perform what has been described as *Global Revision* (Cremmins, 1996). Global revision is the task of supplementing some key sentence, perhaps identified with a sentence extraction tool, with additional information from related sentences. These related sentences from the source document generally lie within the vicinity of the key sentence in the original document. Figure 1 presents such an example.

The summary sentence was written by a professional abstractor.[1] We hypothesise that it was composed by the abstractor on the basis of the three sentences shown below the summary, which we identified manually in the source document. In this example, the abstractor has paraphrased and conjoined the content of the first related sentence with the key sentence via the discourse marker *while*. Additionally, the noun phrase *conflict areas* replaces the phrase *inaccessible areas*. As can be seen by this example, global revisions tend to opportunistically recycle words and phrases from

[1]All text was taken from the United Nations website for Humanitarian Affairs at http://ochaonline3.un.org/humanitarianappeal/index.htm .

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 226–235,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

Summary Sentence:
There are presently an estimated 130,000 internally displaced persons (IDPs) in established camps, while it is estimated that 200,000 people are left behind in conflict areas with inadequate access to food or shelter.

Main Source Sentence:
There are presently an estimated 130,000 internally displaced persons (IDPs) in established camps.

Related Source Sentences:
The condition of an estimated 200,000 people left behind in inaccessible areas with inadequate access to food or shelter is much worse.

Conditions in conflict areas will worsen as the winter storms approach.

Figure 1: An example of global revision.

surrounding text where possible.

Broadly speaking, there are two primary tasks. The first is one of content selection in which, given a set of key and related sentences, the system must automatically identify which phrases and words are important enough to include in the final sentence. Often this is done by ranking words and phrases by some model of their salience. Models for content selection have been studied in previous work on text-to-text summarisation applications. For example, Witbrock and Mittal (1999) presented a model for learning how suitable words from a news article are for inclusion in headlines, or 'ultra-summaries'. In Wan et al. (2003), we presented models akin to that of latent semantic analysis for content selection in which words related to a single main theme were more likely to be selected for inclusion in a summary. Wan et al. (2005) presented methods aimed at recycling dependency relations from the original document, influencing which words are selected for inclusion in the generated summary.

In this paper, we hold the content selection factor constant and assume that one of the content models described above will be employed to rank words and phrases according to their importance. Here, we focus on the second part of the generation problem: namely, that of generating the final ordering of any highly ranked words and phrases. Essentially, the problem becomes one of choosing between the possible combinations of the selected strings, be they words or phrases.

Our baseline approach uses an *n*-gram language model to find the best ordering of words. At a local level, short generated sequences do indeed tend to look like valid word sequences. However, at a global sentence level, *n*-gram generation has no mechanisms to ensure that the sequence is well-formed enough to be grammatical. For example, there is no guarantee that the generated sequence will contain a verb, so it may not even correspond to a sentence.

To address this issue, we present a new method for combining the strengths of an *n*-gram language model (in this paper, we use four-gram models) with a statistical dependency model. Our hope is that grammaticality will be improved by including representations of global sentence structure during a search through different word orders. Our approach is to order an input set of strings (the recycled words and phrases) by attempting to construct a dependency tree that links these strings. This tree is built using a statistical model of dependency relations that is automatically obtained from the Penn Treebank (Marcus et al., 1994).

Our focus in this paper is to describe modifications to Prim's algorithm (Prim, 1957) for finding the maximal spanning tree, which we use to find the best dependency tree. An *n*-gram model is then used to find the best order of sibling nodes in the dependency tree when generating the string yield.

In the remainder of this paper, we begin by contrasting our approach to related work in Section 2. We discuss the Maximum Spanning Tree problem in more depth in Section 3, which presents our use of Prim's algorithm. In Section 4, we describe additional constraints to ensure that the generated tree is a dependency tree. Section 5 describes how to get the final word order once a spanning tree is found. Finally, before we conclude, our preliminary evaluations on the Wall Street Journal sections of the Penn Treebank are presented in Section 6.

2 Related Work

Because of the statistical nature of our work and the lack of symbolic representation as input, traditional text generation (for an overview, see Reiter and Dale 2000) is only peripherally related. Instead, we briefly review work in statistical text generation and paraphrase generation.

Statistical surface realisers (for example, Langkilde and Knight 1998; Bangalore and Rambow 2000) often start with a semantic representation for which a specific rendering, an ordering of words, must be determined. Such sys-

tems usually rank alternative word orderings using a language model to select the best sequence. Our approach blends content selection with surface realisation. That is, words and phrases will only be included in the final sentence if a suitable and natural ordering is possible. In our case, a model based on corpus statistics allows us to define what sounds natural.

Our ultimate goal of text-to-text summarisation is more similar to work on Information Fusion (Barzilay et al., 1999), a subproblem in multi-document summarisation. In this work, sentences presenting the same information are extracted from a set of news articles on the same news event, each perhaps published by a different news company. The basic premise of this work is that what is redundant across sentences is treated as important and worth reporting to the user. This in turn can be supplemented by additional non-repeated information if possible.

At first glance, at least in terms of input and output types (a set of sentences mapped to a single sentence), the Information Fusion and Global Revision tasks seem identical. In fact, it is the inter-relationship of the sentences in the input set that differs between the two problems. Global Revision can be seen as the related single document version of Information Fusion. Crucially, the single document version does not deliver the same type of informational redundancy, and so we need different methods from those used in Information Fusion.

Other text-to-text approaches to generating novel sentences also aim to recycle sentence fragments where possible. More recently, work on phrase-based statistical machine translation has been applied to paraphrase generation (Bannard and Callison-Burch, 2005) and summarisation (Daumé III and Marcu, 2004). Whilst related, such methods benefit from the luxury of paired corpora in which one can learn automatically the transformation between the summary and the original document. What they do have in common with this work is a reliance on language modelling, though predominantly limited to *n*-gram models.

In this paper, we generate word sequences based on a dependency tree which we find using Maximal Spanning Tree algorithms. To our knowledge, this has not been done for text generation. However, the relationship between dependency parse trees and maximal spanning trees has been established in discriminative dependency parsing (McDonald et al., 2005). McDonald et al. specifically examine non-projective dependency trees, whereas this paper examines only projective ones. This difference is important in choosing a spanning tree algorithm.

Additionally, in parsing, one begins with a fixed sequence of words and the search is over tree structures. In contrast, we search for the best word sequence *and* tree structure concurrently, introducing issues that do not arise in parsing. These issues, concerning argument counts of words, are discussed in Section 4.

3 Understanding the Problem

Recall that our input is a set of words and phrases judged to be salient in content. Our task is to use as many as possible in a grammatical ordering corresponding to the novel summary sentence. To begin with, we preprocess the input strings such that each is represented by a single token. Phrases are treated as 'atomic' strings that cannot be broken up further. A phrase is represented by its syntactic head, an annotation that could be obtained automatically via an initial parse of each phrase. Drawing from the earlier example, the noun phrase *conflict areas* would thus be represented by *areas*. This converts the input from a set of strings of arbitrary length into one of single tokens.

Our approach is to find a dependency tree that connects the input head tokens. In searching for the best dependency tree, all possible dependency relations between pairs of words are considered. In effect, this collection of possible edges results in a maximally connected directional graph where each vertex is a word and edges represent a dependency relationship between the two words.

Edges are directional since it matters which word is the head and which is the child. Edges also have automatically calculated weights that represent the strength of the relation. Thus, an edge that is seemingly implausible as a dependency relation will have a low weight. In contrast, an edge representing a dependency relation that one observes frequently receives a high weight.

Finally, in this particular graph of possible dependency relations, we keep track of the relative ordering of the head and its child: either the head occurs to the left of the modifier in the word sequence, or to the right.

We base our calculations of dependency rela-

tion strength on the models proposed by Collins (1996). Our version of the models differs in that we do not account for distance between head and modifier words. We define a dependency relation event as having the following three attributes: the head token, the word token and its relative ordering, which can be *leftwards* or *rightwards*. Thus, the probability of the relationship for two words A and B in a direction D is defined as follows:

$$prob(A,B,D) = \frac{cnt(rel(A,B,D))}{cnt(co(A,B))}$$

where the frequency of seeing that relation is normalised by $co(A,B)$: the number of times we saw those words together in a sentence, i.e., a co-occurrence count.

We can back off to part-of-speech tags in two stages as Collins describes. The first stage is to alternate between using the part of speech for one word in the pairing:

$$prob_{1pos}(A,B,D) = \frac{cnt(rel(A_{pos},B,D)) + cnt(rel(A,B_{pos},D))}{cnt(co(A_{pos},B)) + cnt(co(A,B_{pos}))}$$

The final backoff stage is to use only parts-of-speech:

$$prob_{2pos}(A,B,D) = \frac{cnt(rel(A_{pos},B_{pos},D))}{cnt(co(A_{pos},B_{pos}))}$$

A combination of these backoff probabilities is done linearly:

$$prob_{backoff1} = \lambda_1 \times prob(A,B,D) + (1-\lambda_1) \times prob_{backoff2}(A,B,D)$$

where $prob_{backoff2}$ is in turn defined as:

$$prob_{backoff2}(A,B,D) = \lambda_2 \times prob_{1pos}(A,B,D) + (1-\lambda_2) \times prob_{2pos}(A,B,Direction)$$

where:

$$\lambda_1 = \frac{cnt(co(A,B))}{cnt(co(A,B)) + 1}$$

$$\lambda_2 = \frac{cnt(co(A_{pos},B)) + cnt(co(A,B_{pos}))}{cnt(co(A_{pos},B)) + cnt(co(A,B_{pos})) + 1}$$

```
01 add all vertices into RemainingNodes
02 BestEdges.store(root, ⟨null,root⟩)
03 loop while BestEdges not empty
04     currentEdge = max_e (weight(e) : e ∃ BestEdges)
05     BestEdges.remove(currentEdge)
06     RemainingNodes.remove(modifier(currentEdge))
07     addEdgeToTree(currentEdge)
08     ∀ E=⟨ modifier(e), v ⟩ : v in RemainingNodes
09         E_best = BestEdges.retrieve(v)
10         if weight(E_best) < weight(E)
11             BestEdges.store(modifier(E), E)
```

Figure 2: Prim's algorithm for Generating Dependency Trees

3.1 Using Prim's Algorithm

Ultimately, what we want is a dependency tree that contains all the nodes in the graph. However, there are numerous trees that might possibly provide such an appropriate structure. One method of defining an optimum tree is to score trees based on some function of the weights of its component edges and then rank them to choose the best one.

Finding a spanning tree that is maximal in the sum of its edge weights is a well studied problem in the general area of graph theory. One such solution is Prim's algorithm, which finds the exact solution by consecutively adding the best possible edge until all words are part of the tree. This should provide a projective dependency tree in which none of the dependency edges cross given the word sequence yield of the tree.

Our initial version of Prim's algorithm for finding optimal dependency trees is presented in Figure 2. We now step through the pseudo-code with a description outlining the intuitions for choosing this algorithm.

3.1.1 Intuitions Underlying the Algorithm

To begin with, a dummy root token is added to the spanning tree (line 2). The process starts by finding edges containing words that are likely to be the head of the main clause (line 8), namely those that have been observed in the corpus that have a relationship with the dummy root. These will be predominantly verbs. For these words, no alternative edges connecting them to the tree yet exist so we store them (line 11).

The best of the root-to-vertex edges is chosen on the next pass of the loop (line 4) and added to the spanning tree (line 7). From the newly appended word (likely to be verb), a larger subset of words is now reachable. In fact, due to the smoothed de-

pendency model, all remaining words are considered as modifiers. That is, most relationships will have a non-zero weight. If this new edge beats any previously viewed edge to this word, we keep it and discard the worst one (line 10, 11). In this last step, we hope that words which previously had a tenuous attachment point to the tree, due to an improbable dependency relation, will now find a better head word to attach to.

Note that, when examining new edges in line 8, we consider dependencies in a leftward direction as separate from rightward relations. If both exist, then both are compared in line 10. In the end, only one alternative is kept depending on the strength of its weight.

The process loops until all nodes have been added to the spanning tree.

3.1.2 Hard Constraints

To complete the story, we used two additional heuristic filters which impose hard constraints on the tree structure. For simplicity, these are not listed in Figure 2. Because of our smoothing method, it is possible for words that are not verbs to end up attaching the root token. Since this is almost never the case, in this step we only considered words whose part-of-speech is 'verb' for this position.

The second constraint disallows a noun from modifying another noun. This rather brutal filter is an artifact of our experimental scenario in which nouns often represent full phrases. Adjoining a noun phrase to another can make the resulting string ungrammatical in this context.

4 Accounting for Argument Counts

Unfortunately, in the context of statistical generation, a maximal spanning tree isn't necessarily a well-formed dependency tree. The model of dependency relations and the search strategy based on Prim's algorithm treat dependency triples as independent events. However, this is not the case. One simple way in which the events are clearly not independent is that words have a minimum and maximum number of permissible modifying arguments as dictated by their grammatical category.

For example, prepositions generally take one and only one modifier in which the modifier points leftwards to the preposition. Similarly, a main clausal verb usually requires a subject as a minimum but could take on any number of additional arguments including, for example, a potentially endless list of adverbs. Resulting trees could thus have words missing arguments and words with too many arguments.

4.1 Argument Count Error Cases

We analysed the algorithm with several examples to see how such errors were being made. Three error cases were identified, which we have dubbed: Theft of a Good Argument; the Greediness Problem; and A Shortage of Words.

4.1.1 Theft of a Good Argument

In this particular case, the algorithm dictates that the newly attached node in the spanning tree should offer an attachment point to each of the remaining unattached nodes. Consider the case in Figure 3. The node $word_a$ might well be the only possible subject noun left for the verb. However, it may be the case that:

$$prob(word_b, word_a, D) > prob(verb, word_a, Left)$$

in which case, the node $word_b$ will steal away the much needed subject argument (line 10). In general, a word may be the last remaining possible argument filler for an obligatory argument slot of a parent word. As we have already considered edges leaving the original parent word at the time of its attachment to the tree, no further potential argument fillers will be examined. Thus, once stolen from, there may be no chance to fill obligatory argument slots.

4.1.2 The Greediness Problem

Figure 3 indicates that the node $word_b$ now has two nodes that could potentially attach to it. These may be the final attachments made to the spanning tree if they are the last two remaining nodes. However, if $word_b$ were in fact a word with limited argument slots, such as a preposition, it would now be over-saturated, which will ultimately result in an ill-formed dependency tree. In general, this may occur at any level of the tree if we greedily attach arguments to inner nodes.

4.1.3 A Shortage of Words

Conversely, some nodes may end up with insufficient arguments. This could happen at an inner node higher up in the tree if its arguments are stolen. It could also happen at the leaf level. If the last node to be attached to the spanning tree should have been an inner node, there will be no more words left to fill its argument slots. This might

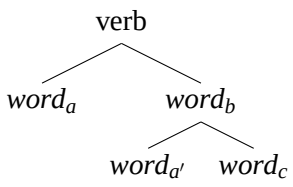


Figure 3: Argument error cases. $word_{a'}$ is 'stolen' node

happen if, for example, a preposition was the last word to be attached and it was not part of some phrasal verb.

4.2 A Model of Argument Counts

The model of argument counts and its usage in our modification of Prim's algorithm is designed to address the first two error cases. The third is left to future work since we cannot backtrack with this search algorithm. The model simply keeps track of the number of times a given word W has n modifiers, again accounting for leftward or rightward directions.

Given a word W, and an argument count n in the direction D, its probability is defined as:

$$prob_{arg}(W,n,D) = \frac{cnt_{arg}(W,n,D)}{cnt(W)}$$

We can compute a model that backs off to part-of-speech in an analogous way to the model defined for dependency relations.

$$\begin{aligned} prob_{argbackoff}(W,n,D) = \\ \lambda \times (prob_{arg}(W,n,D)) \\ +(1-\lambda) \times (prob_{arg}(W_{pos},n,D)) \end{aligned}$$

Again, lambda is defined in terms of the denominator $cnt(W)$:

$$\lambda = \frac{cnt(W)}{cnt(W)+1}$$

4.3 Modifications to Prim's Algorithm

In our approach based on Prim's algorithm, attachment is final and thus only methods that attempt to prevent construction of an ill-formed tree are possible. That is, tree repair is not currently a part of our algorithm.

Given that Prim's algorithm does not allow backtracking, we have to re-weight edges using the model for argument counts during the search to avoid potential traps that would lead to an error case. Specifically, the argument count model is used in two places. Updating the priority queue to account for changing status in argument saturation attempts to combat the Greediness Problem. In choosing the best edge to keep, we can also avoid Thefts of a Good Argument.

Note that such a scheme would affect the edge weights dynamically as the search progressed. Since we rescore edge weights during search, they are obviously not fixed before the search begins, and thus Prim's algorithm is no longer guaranteed to produce the globally optimal solution.

4.3.1 Rescoring the Priority Queue

Recall that the BestEdges priority queue is checked at the beginning of the main loop of the algorithm to choose the next word to attach to the spanning tree. To avoid over-saturating a node, we adjust the 'attractiveness' of a dependency relation if its head word has peaked in terms of its optimum valency, possibly due to previous attachments.

To do so, we keep track of an argument count per word. When a modifier is added to the tree, the argument count for the head word is incremented by one. All remaining edges in the priority queue that attach to that head word must then be adjusted to account for this increment.

Re-weighting edges in BestEdge is computed as follows:

$$\begin{aligned} prob_{adjust}(H,M,D) = \\ prob_{argbackoff}(H,n+1,D) \times prob(H,M,D) \end{aligned}$$

where n is the number of current children in direction D that the head H has in the spanning tree.

Typically, after the optimum number of edges for a word, additional edges are less probable or in fact ungrammatical, and the edge will correspondingly be adjusted to receive a lowered score. Once edges have been adjusted and the queue is sorted, the best edge is chosen and its modifier is attached to the spanning tree.

An additional advantage to re-weighting an edge in the priority queue is that it allows that edge to be more readily replaced with one that attaches the modifier to a head that still has available argument slots.

4.3.2 Choosing the Best Edge

When comparing alternative attachment points (line 8), we must choose between two competing parent nodes offering attachment points to the modifier in question. At the time of this comparison, argument thefts can also be avoided if we

consider how many argument fillers the old parent still has. These are edges in the BestEdges priority queue that attach to the parent.

A new competing edge (with head H and modifier M) is re-weighted and given a lower weight if it would lead to a Theft of a Good Argument condition. The priority queue BestEdges stores P, the best attachment parent node seen so far for the modifier M. We must identify the potential for the valency of the old parent node, P, to be satisfied if M were stolen from it. Furthermore, the direction of its relationship, leftward and rightward, must also be accounted for such that there is a satisfaction score for each of these directions.

To do so, we find all edges in BestEdges with P as a head. These are divided into leftward and rightward sets, L and R. We first define the 'potential' for P to be satisfied. This is simply the arithmetic mean of the probabilities that W has 0 to m children, where m is either the size of L or R minus 1 respectively.

$$prob_{satisfied}(W,m,D) = \frac{\sum_{i=0}^{m} prob_{argbackoff}(W,i,D)}{m}$$

This allows us to re-weight the new competing edge (with head word H and modifier M in direction D) as:

$$prob_{re\text{-}weight}(H,M,D) = prob_{satisfied}(H,n,D) \times prob(H,M,D)$$

We then continue the comparison at line 10 with this re-weighted edge. Thus, if an old parent will not be satisfied after the modifier is stolen, then the new edge is penalised. If the parent node may still be satisfied, then the new edge is not disadvantaged.

5 Generating a Word Sequence

Once the tree has been generated, all that remains is to obtain the ordering of words based upon it. Because dependency relations in the tree are either of leftward or rightward direction, it becomes relatively trivial to order child nodes with respect to the parent. The only difficulty lies in finding a relative ordering for the leftward children, and similarly for the rightward children.

We propose two alternatives for finding the relative ordering of sibling nodes. The first is based on the strength of the dependency relation. This method sorts sibling nodes in descending order based on the final weight of the dependency relation as it was added to the tree. That is, the child node with the strongest dependency relation is placed closest to the parent node. For example, for leftward siblings, the relation with the highest edge comes just after (on the right of) the parent node. We will refer to this as a *Relation-based Ordering*.

The second method uses an n-gram language model to score word sequences between the strings yield of each sibling node. A greedy algorithm is used to determine the order of the phrases. For the set of rightward relations, the algorithm chooses the best word or phrase to follow a start-of-sentence marker based on n-gram probabilities. Once the best phrase is chosen to follow the start-of-sentence token, it then becomes the new point of comparison. Of the remaining phrases and words, we select the one that has the best n-gram probability following the most recently chosen phrase (or word). This continues until all input elements are consumed.

Where possible, an overlap window of size six ($2n-2$) is examined, split at the point of the concatenation of the two phrases. That is, the last three words of the left phrase and the first three words of the right phrase. The probability of the concatenation is simply the probability of this substring which is in turn the product of the probabilities of four-grams within that substring. We will refer to this as a *N-Gram Ordering*.

Similarly, for the leftward relations, we use the greedy generation algorithm. However, instead of using the start-of-sentence marker, we begin with the phrase that realises the head node.

6 Evaluation

6.1 Task

In planning our evaluation, we sought to choose a task for which issues in content selection could be temporarily put on hold while we instead focused on appraising our approach for finding a grammatical ordering of input words and phrases. Thus, we simplified our task to that of ordering words and phrases from a single sentence.

As a surrogate measurement of grammaticality, we tested to see if we could regenerate the original string. In this task, manually delimited noun phrases and other words from a sentence are provided out-of-order as input to the generation mod-

ule. The aim is to see how much of the original sentence can be regenerated. This can be measured by any string comparison method. In our case, we used the Bleu metric (Papineni et al., 2002).

This is actually an overly strict evaluation. A set of input words and phrases could have more than one grammatical ordering. If we were to generate one of the alternative orderings, our score would be adversely affected.

6.2 Training and Data

As we needed an annotated corpus of dependency relations, we used the Wall Street Journal portion of the Penn Treebank to obtain our dependency related frequencies. Dependencies were sourced from the events file of the Collins parser package which is in turn a compilation of the dependency events found in training sections 2-22 of the corpus. Development was done on section 00 and testing was performed on section 23.

A four-gram language model was also obtained from the same training data. In both baselines and in our system, we smooth the data using Katz's method which backs off to smaller sized *n*-grams.

As input, we used the tagged version of the WSJ sections which includes not only part-of-speech tags for each word but also delimitations on the base noun phrases, that is, noun phrases without nested constituents. We assumed that the last word of a base noun phrase is the head word of that phrase. Thus our input set of strings per sentence were the heads of base noun phrases and intervening single words.

6.3 Baselines

We decided to compare our system with two baselines. The first is the greedy *n*-gram ordering algorithm used to order rightward siblings in the dependency tree. Again, a start-of-sentence token is used and where possible a maximum overlap window of size six is used. As input, this baseline is given the same set of base noun phrases and words as our system. Noun phrases are not reduced to a single head word for this baseline generator. Thus, it too will receive the same benefits of word overlap due to a re-use of noun phrases. Note that this algorithm simply orders the input based on a language model and makes no use of tree structure.

When concatenating pairs of single tokens, the first baseline is disadvantaged because the maximum overlap window is limited to just a bigram.

System	Bleu
Tree	0.296
Tree+*N*-Grams	0.307
Tree+Arg	0.300
Greedy baseline	0.279
Viterbi baseline	0.107

Figure 4: Bleu scores

To account for this, we search with a full four-gram model over single tokens using a Viterbi-like algorithm with a beam of 100. As input, base noun phrases are also broken up into component words. Thus, this baseline tries to regenerate the test sentence based only on individual tokens, and does so without knowledge of noun phrase boundaries.

6.4 Results

We obtained favourable results, which indicate that an approach that orders words by first trying to build a dependency tree does produce more fluent text, as measured by Bleu. We present the Bleu scores in Table 4. The system named 'Tree' uses the simple version of Prim's algorithm presented in Figure 2 where sibling nodes are ordered based on the strength of their relations. The 'Tree+*N*-Gram' system differs only in that a language model and the greedy *n-gram* ordering algorithm are used to order siblings. The 'Tree+Arg' system uses the modifications for handling argument counts on top of the 'Tree+*N*-Gram' system.

Using the sign test and the sampling method as outlined in Collins et al. (2005), we tested for statistical significance. All differences are significant at $p < 0.01$ except between the variations of our systems. Although using *n*-grams to order sibling nodes obtained a slightly higher score, the lack of significance might be attributable to the fact that the dependency structure is responsible for most of the improvements in Bleu score and that the language model, when used in this manner, does not add much. Thus, in hindsight, it makes sense that we are less able to observe the effects of using the language model.

It also helps to examine the text that is actually generated by each system. In Figure 5, we present examples of typical generation cases for both the Tree+Arg system and the Greedy baseline.

Tree+Arg: a developer of mr. shidler 's company specializes in l.j. hooker and mr. simpson claims is in to have $ 1 billion a former senior executive commercial real-estate investment and assets

Greedy: in of is to have $ 1 billion in a former senior executive and commercial real-estate investment and claims assets a developer l.j. hooker specializes mr. simpson ; mr. shidler 's company

Original Sentence: mr. shidler 's company specializes in commercial real-estate investment and claims to have $1 billion in assets; mr. simpson is a developer and a former senior executive of l.j.hooker

Figure 5: Example generated sentences

6.5 Discussion and Future Work

Overall, we observed an improvement in the degree of string regeneration which we interpret as an improved ability on the part of our system to generate grammatical strings.

It is worth reflecting on the limitations of our approach both when the process operates as intended as well as when it performs poorly. By using corpus based dependency probabilities, we are biasing the system to recreate the semantic content of the corpus, insofar as our dependency representations are capable of capturing such information. Thus, in the evaluation, if the unseen test sentence which we want to regenerate should differ drastically in semantics from previously seen data, it will be difficult to recreate this sentence, even if the algorithm otherwise works as intended. For example, if a noun phrase that was always seen in the object position is suddenly in subject position in the test sentence, we will have little chance of regenerating the complete sentence. However, both our approach and our *n*-gram generation baselines would suffer from this.

Since we almost never recreate the original sentence completely, it is clear that our algorithm can still be improved. Considering the bigger picture, we have in effect shoe-horned Prim's algorithm into serving as an algorithm for finding an optimum dependency tree. As already discussed, dynamically weighting edges to account for satisfaction of argument counts results in the problem becoming more complex. Our modifications to handle argument counts seem still to be insufficient overall to avoid invalid dependency trees being considered.

Disappointingly, our modifications for handling argument counts seemed to have only a slight effect. One key limitation in the approach is the lack of a backtracking facility in the search algorithm. As noted above, at any given point in the search, what might have been the best edge to some node may suddenly become obsolete as a consequence of adding some other node to the tree. One option would be to keep an *n*-best list of edges instead of using a max function. We are currently working on such modifications to test for any improvements. It should be noted, however, that keeping an *n*-best list increases the complexity. We are also working to improve our model of argument counts further.

In our evaluation, noun phrases were conveniently delimited in the WSJ corpus. We intend to explore methods to either automatically delimit noun phrases or else to generate them from scratch. Finally, our evaluation in this paper focused only on string regeneration and used Bleu as a metric. We will also examine other metrics, such as the one proposed in Mutton et al. (2007), which evaluates the grammaticality of novel sentences. Additionally, we hope to evaluate our work in the context of the original task of global revision in abstract-like summarisation.

7 Conclusion

In this paper, we presented a mechanism for generating a novel sentence from a predetermined set of words. We argued that such a mechanism may be used for performing global revision as part of a larger text-to-text generation system for producing abstract-like summaries. Our approach employed a variation of Prim's algorithm to construct a dependency tree to be used in conjunction with a language model for ordering a set of words. The construction of the tree was performed using a statistical model of dependency relations. Our hope was that a language model would improve word order at the local level, whilst a syntactic model would enforce a valid grammatical structure at the global level. We observed a significant gain in performance over our baseline in a string regeneration task, attaining an system Bleu score of .307. We conclude that an approach that combines both syntactic and *n*-gram information can benefit from both types of information.

8 Acknowledgements

This work was funded by the Centre for Language Technology at Macquarie University and by the

CSIRO Information and Communication Technology Centre. We would like to thank both groups as well as the reviewers for useful comments and feedback.

References

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th Conference on Computational Linguistics*. Universität des Saarlandes, Saarbrücken, Germany.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, Michigan.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th conference on Association for Computational Linguistics*, Morristown, NJ, USA.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Morristown, NJ, USA.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*. San Francisco.

Edward T. Cremmins. 1996. *The Art of Abstracting*. Information Resources Press.

Hal Daumé III and Daniel Marcu. 2004. A phrase-based hmm approach to document/abstract alignment. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

Hongyan Jing and Kathleen McKeown. 1999. The decomposition of human-written summary sentences. In *Research and Development in Information Retrieval*, pages 129–136.

Irene Langkilde and Kevin Knight. 1998. The practical value of N-grams in derivation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, New Brunswick, New Jersey.

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

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.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Morristown, NJ, USA.

Andrew Mutton, Mark Dras, Stephen Wan, and Robert Dale. 2007. Gleu: Automatic evaluation of sentence-level fluency. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, USA.

R. C. Prim. 1957. Shortest connection networks and some generalisations. *Bell System Technical Journal*, 36:1389–1401.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.

Stephen Wan, Mark Dras, Cécile Paris, and Robert Dale. 2003. Using thematic information in statistical headline generation. In *The Proceedings of the Workshop on Multilingual Summarization and Question Answering at ACL 2003*. Sapporo, Japan.

Stephen Wan, Robert Dale Mark Dras, and Ccile Paris. 2005. Towards statistical paraphrase generation: preliminary evaluations of grammaticality. In *Proceedings of The 3rd International Workshop on Paraphrasing*. Jeju Island, South Korea.

Michael J. Witbrock and Vibhu O. Mittal. 1999. Ultra-summarization: a statistical approach to generating highly condensed non-extractive summaries. In *Proceedings of the 22nd annual international ACM SIGIR conference*.

Modelling and Exploring the Network Structure of Terminology Using the Potts Spin Glass Model

Kyo Kageura
Library and Information Science Course
Graduate School of Education
University of Tokyo
kyo@p.u-tokyo.ac.jp

Takeshi Abekawa
Library and Information Science Course
Graduate School of Education
University of Tokyo
abekawa@p.u-tokyo.ac.jp

Abstract

Terminologies of any domain contain a substantial number of compounds or multi-word terms. Through the shared constituent elements of the compounds, terms form complex networks, which constitute a terminology. A variety of methods developed for the analysis of complex networks can therefore be used for exploring the structure of terminologies. In this paper we will apply a theoretically valid method for detecting communities in networks to the partitioning of terminologies or clustering terms, in order to explore the structure of terminologies. Experiments show the usefulness of the method for analysing the motivated structure of terminologies.

1 Introduction

Terminologies of any domain contain a substantial number of compounds. In Japanese terminologies, for instance, about 85% of the terms are compounds (Nomura and Ishii, 1988). Terms are thus connected with one another through shared constituent elements and form complex networks which constitute the terminology. The exploration of these networks, which reflect the motivatedness of terms and show the associative structure of terminologies, is an important theoretical issue for the formal analysis of terminology and of lexicons in general, with potential applications such as dictionary compilation, postprocessing of automatically extracted sets of terms, etc. It is complimentary to the description and processing of the syntagmatic aspects of complex terms (e.g. Daille, et. al., 1996; Jacquemin, 2001) as well as to text-oriented terminology studies, including such applications as term extraction and clustering (e.g. Heid, 1999; Weeds, et al., 2005). While these latter types of studies have been actively pursued, the associative structure of terminologies has not been studied to the same extent, though important exceptions exist (Hamon and Nazarenko, 2001; Grabar and Zweigenbaum, 2004).

This paper explores the complex networks of terminologies, by applying a network community detection method based on statistical mechanics to carry out a divisive clustering or partitioning of terminologies (Scott, 2000). Out theoretical stance, which is described in section 2, is that there is a theoretical precedence of terminologies as a set over individual terms occurring in texts (Kageura 2002), and thus we need to explore the structural nature of terminologies fully, in such a way that the exploration methods properly reflect theoretical views on and the interpretative framework of terminologies, and enable us to qualitatively interpret the results. We argue for the appropriateness of a Potts spin glass model-based divisive clustering of complex networks as a theoretical model for exploring and visualising (not in the sense of visualisation but in the sense of exploring the hidden structure of) the terminological networks.

In the following, we will first clarify the theoretical status of terminological networks constructed by common constituent elements in the study of terminology. We will then argue that to explore these associative terminological networks, a divisive clustering method based on the Potts spin glass model allows a valid interpretations of the results. After applying the method to the mixed terminologies of three domains and observing its performance from the point of view of divisive

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 236–245,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

Table 1. A putative terminology consisting of 12 terms

"text classification"	"document classification"	"information retrieval"	"library classification"
"disease diagnosis"	"document information"	"medical information"	"medical diagnosis"
"medical aid system"	"automatic text classification"	"diagnosis record"	"disease image record"

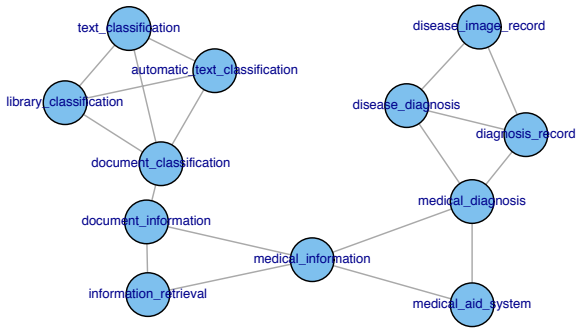


Figure 1. A network constructed from the sample terminology in Table 1

clustering, we will analyse, both quantitatively and qualitatively, the characteristics of the network of documentation terminology, contrasting it with the network of documentation and artificial intelligence terminologies combined.

2 Terminological networks

Take, for instance, a putative terminology consisting of 12 terms (Table 1). This terminology forms the network shown in Figure 1. Formally, this network is an undirected weighted graph, with vertices representing terms and edges representing shared constituents between terms. Naturally:

$$\text{degree}(v_i) \simeq \sum_j \text{frequency}(c_{ij}) \quad (1)$$

$$\text{weight}(e_{ik}) = |\{c_{ij}\} \cap \{c_{kl}\}| \quad (2)$$

where v_i is the vertex or term with index i, c_{ij} is a j-th constituent of the term representing v_i, and e_{ik}, which is defined by the number of common constituent elements between the two terms v_i and v_k, is the edge between v_i and v_k. As we will see in section 4, even for a small- or medium-sized terminology consisting of one to two thousand terms, this network becomes very complex with tens of thousands of edges. As the degrees of vertices reflect the frequency distribution of constituent elements in the terminology, which follows the power law (Kageura, 1998), the network has a scale-free nature (Albert and Barabasi, 2002).

These networks have a theoretical status in the study of terminology. Firstly, we observe that terminology as a set or terminological space, as opposed to individual terms, constitutes an object of study in its own right. It is the concept of terminology as a set that enables us to recognise individual terms as empirical objects. Unlike sentences, terms cannot be arbitrarily constructed by individuals; terms become terms only when they are accepted by a community and become part of a terminology. This process is not determined by any person or group. This, combined with the fact that complex terms show a strong inclination towards motivatedness vis-à-vis *the overall terminology* (Nomura and Ishii, 1988; Sager, 1991), supports the argument not only for the theoretical importance but also for the reality of terminological space which is socially constructed and shared. This in turn is internalised — unlike individual terms, not consciously — in subject-specialists' minds. It is a terminological equivalent of *the lexicon* — a concrete social product — that belongs to the Saussurian *langue* (Saussure, 1910-1).

This terminological space is governed by the conceptual system or field of the domain as well as by the system of linguistic representations of concepts (Sager, 1991). Terms are materialised within this terminological space, guided by the conceptual system and using linguistic items (Rey, 1985). The process can be postulated as: (i) concepts to be represented are identified within the overall conceptual system; (ii) in accordance with the nature of concepts to be materialised as terms within the relevant conceptual subsystem, linguistic items representing required conceptual features are consolidated; (iii) terms are formed using these linguistic items, following the dictates of the conceptual subsystem and the system of linguistic representation of concepts; and (iv) the terms, when appropriate, are incorporated in the terminology. The standard description of the formation of terms by means of the relation between a term and its constituent elements belongs to stage (iii). Within this model, a well-structured terminology with many motivated terms can be seen as constructed within a well-coordinated system of concepts and linguistic representations, and a less structured terminology with fewer motivated terms within a less

systematic or 'hot' and chaotic system of concepts and linguistic representations. Note here that there is a general correlation between the systematicity of concepts and the systematicity of linguistic representations.

Correspondingly, the network structure defined over the terminology through the shared constituents is not merely a convenient, practical or opportunistic relational representation of a group of individual terms. Rather, to the extent that the terminology of a domain reflects the nature of a conceptual system through its relatively motivated and partially systematic linguistic forms of terms, the terminological network thus constructed reflects the conceptual system that underlies the terminology and the system of linguistic representations. Detecting communities in terminological networks is therefore a means to explore these dual systems of concepts and linguistic representations.

3 Community detection algorithm

3.1 State of the art

A number of community detection or clustering methods have been proposed and used to date (throughout this paper we use 'community detection' and 'clustering' synonymously; we also use 'partitioning' when we emphasise the divisive aspect of community detection). The methods can be divided into divisive/partitional and agglomerative approaches (Jain, et al., 1999; Newman, 2004; Scott, 2000). Within our context, a divisive approach is the natural choice, because it regards the terminology as a whole as the basic object of study, while agglomerative approaches — including relatively new methods (Handl, et al. 2003) — regard individual terms as the basic object.

With the growth in research into complex networks (Albert and Barabasi, 2002; Masuda and Konno, 2005; Newman, 2003; Strogatz, 2001; Watts and Strogatz, 1998), a wide variety of divisive algorithms have been proposed and examined recently (Duch and Arenas, 2005; Fortunato, et al., 2004; Newman, 2004; Newman and Girvan, 2004), in addition to a traditional minimum cut algorithm (Cheng and Wei, 1991; Fujishige, 2002). From the point of view of clustering applications, it goes without saying that the method that performs best vis-à-vis an ideal result should be chosen. So the comparative examination of different methods has been carried out using common test data (e.g. Danon, et al., 2005). In clustering applications, the performance and the upper bound differs, depending on the extent to which the necessary features are present. The present framework, on the other hand, is different from that of common clustering applications. Firstly, it is common knowledge in the field of terminology that the structure of each underlying system of concepts is multifaceted and multidimensional, that the coherency of the conceptual system varies from one terminology to another, and that the system of linguistic representations is not totally motivated. The first two characteristics lead to ambiguity concerning what are 'ideal' clusters, which in turn leads to deconstruction of the concept of a 'gold standard'. The third characteristic means that the features are only partially provided. In addition to these, as the essential aim of using divisive clustering in this study is to employ it as a guide to the qualitative diagnosis of the systematicity of concepts and the motivatedness of linguistic representations, the method to be adopted should fit the basic theoretical and interpretative framework of the universe of terminological networks. This contrasts with the quest for the method with the best clustering performance vis-à-vis some 'gold standard'. Incidentally, applying graph-based methods to word clustering is not new (Aizawa and Kageura, 1998; Matsuo, et al., 2006), but this prior work is application-oriented and uses cooccurrences of words.

3.2 Clustering using the Potts spin glass model

After examining the interpretative relevance of various methods (Cheng and Wei, 1991; Duch and Arenas, 2005; Fortunato, et al., 2004; Fujishige, 2002; Newman, 2004; Newman and Girvan, 2004), we adopted the algorithm developed by Reichardt and Bornholdt (2006), which is based on an infinite range Potts spin glass model (Dorogovtsev, et al., 2004; Nishimori, 1999). A similar model is used in NLP by Takamura, et. al. (2005) for extracting the semantic orientations of words. We found that this method fits our interpretative framework admirably. But before presenting our argument for adopting this method, let us first briefly describe the Potts spin glass-based method.

Formally, the Potts model consists of a lattice of N sites, on each of which is placed a spin that can take q-states. The Hamiltonian or energy function H (which is used as a cost function to be min-

imised in the algorithm) of the Potts model is given as:

$$H(\{s\}) = -\sum_{(i,j)} J_{ij}\delta(s_i, s_j)$$

where s_i and s_j represent the state of spins i and j, and J_{ij} represents the strength of mutual influence between s_i and s_j. δ gives 1 when s_i and s_j are in the same state and 0 otherwise.

Here $\{s\}$ follows the Boltzman distribution:

$$P(\{s\}) = \frac{\exp(-\beta E(\{s\})}{\sum_s \exp(-\beta E(\{s\})},$$

where the denominator is the normalisation factor called the partition function, E gives the energy and β is a constant called inverse temperature.

Starting from the assumption that the Hamiltonian or the quality function for detecting communities in a network should: (i) reward internal edges between vertices of the same group (or in the same spin state); (ii) penalise missing edges between vertices in the same group; (iii) penalise existing edges between different groups; and (iv) reward nonlinks between different groups, and with a few simplifying assumptions, Reichardt and Bornholdt (2006) proposed the following Hamiltonian:

$$H(\{\sigma\}) = -\sum_{i \neq j} (W_{ij} - \gamma p_{ij})\delta(\sigma_i, \sigma_j),$$

where W_{ij} denotes the adjacency matrix of the graph with normalised weight for edges, p_{ij} denotes the null model that gives the baseline probability that a link exists between node i and j (in actual applications, we chose the uniform distribution for p_{ij}), and γ is the parameter for distributing the weight between the reward for internal links and the penalty for internal nonlinks. This represents a spin glass with mutual influence $j_{ij} = W_{ij} - \gamma p_{ij}$.

This Hamiltonian can be transformed to the modularity Q defined by Newman and Girvan (2004), which compares the actual and expected fractions of links between vertices in the same group:

$$Q = \sum_i (e_{ii} - a_i^2)$$

where e_{ii} indicates the actual fraction of links that connect vertices in the group i, and a_i^2 indicates the corresponding expected value. The modularity takes a value between 0 and 1. The higher value corresponds to a lower Hamiltonian and *vice versa*. We will later use the modularity instead of H as a top-down quality index of community detection.

The solution of the divisive clustering is given by the assignment of spins in the *ground state* of the system. Reichardt and Bornholdt (2006) give a computationally tractable method of finding the ground state by simulated annealing, based on the heat bath update rule at a given temperature. An R library as well as a C library with a Python interface that carry out the computation is made available by Csardi (2006). We used R library for the analyses.

Defining the problem of community detection as finding the ground state for the Potts spin glass model gives a straightforward interpretation in terms of the systematicity of terminology. The ground state, in which the Hamiltonian is minimised and the modularity is maximised, is the most 'natural' state that appears when the 'temperature' of the system has cooled down. In the case of a terminological network, the physical concept of 'low energy' reflects a state in which the maximally motivated structure is retained. As such, quite apart from the results of the clustering, the minimum value of the Hamiltonian (or correspondingly, the maximal value of the modularity) can be interpreted as indicating the basic motivatedness of the terminology. The resultant clusters can be interpreted as the basic and maximally motivated state of the terminology when the contribution of 'excited' states is suppressed.

In short, we obtain not only the clustering result for each terminology network, but also an index that reflects the basic degree of systematicity or motivatedness of the terminology. Given that any terminology is only partially motivated, which, from the point of view of clustering, means that the features are not completely given and the upper bound of the clustering results cannot be determined with 100 per cent validity, having some parameters that correspond to the natural interpretation of the object under observation is particularly important for analysing the terminological data — more important than choosing a method that happens to give the highest performance vis-à-vis some sort of 'gold standard'. The meaning of the Hamiltonian given in the Potts spin glass model and the correspondence between the Hamiltonian and the modularity provide us with such a parameter.

Table 3. Basic quantities of the mixed terminological data (plus DC)

	# terms	# dupl. terms	# const.	# dupl. const.	# edges	subgraphs max	6	5	4	3	2	1
DC	1228	–	831	–	11251	1046	–	–	2	2	8	151
DC+AI	2918	35	1823	306	36479	2571	–	2	3	6	11	285
AI+GG	3502	3	2508	145	34299	2857	1	1	1	10	33	534
GG+DC	3008	0	2085	101	27521	2405	–	–	1	9	28	516
ALL	4695	38	3001	483	54426	3942	1	2	2	12	30	633

This method also provides the following additional benefits: (i) it allows us to control the number of communities or clusters by using the parameter γ or by explicitly setting the maximum number of spin states; (ii) it allows us to detect communities hierarchically, though we will not explore this aspect in this paper; and (iii) the distribution of the sizes of the detected communities is not overtly skewed, unlike the decomposition carried out by such methods as minimum cuts. In relation to point (iii), we also applied minimum cut algorithms to the experimental data introduced below. As the terminological networks have a scale-free nature, the minimum cuts only cut out small groups and do not give desirable results, which is in conformity with Cheng and Wei (1991).

4 Clustering experiments

4.1 Experimental setup

In order to obtain a general idea of how the Potts spin glass model-based divisive clustering works, we first carried out an experiment in which the performance of clustering could be quantitatively observed. For the experiment, we used Japanese terminologies of three domains as basic data: documentation (DC) (Wersig and Neveling, 1984), artificial intelligence (AI) (JSAI, 1990), and geography (GG) (JME, 1981). The sources for these terminologies were standard and/or widely used reference lists used from the 1980s to 1990s. As a domain, DC and AI are close to each other, while GG is distant from the other two. The basic quantities of the terminological data are given in Table 2.

Table 2. Basic quantities of the data

	# terms	# const.	# edges	# isolates
DC	1228	831	11251	134
AI	1725	1298	33345	152
GG	1780	1355	16239	382

In order to quantitatively evaluate the nature of the method from the point of view of clustering performance, we carried out an experiment that involved re-partitioning mixed terminological networks made from pairwise combinations of the terminologies, as well as a combination of all three terminologies.

In constructing the terminological networks, we did not consider symbols, numbers or 25 functional elements such as “の” and “化”. We did not distinguish heads from modifiers, for two main reasons: (1) as we are concerned with the conceptual system and the terminological space rather than the syntagmatic aspects of individual forms or formation of terms, we deliberately put aside this distinction as the first step; and (2) from the point of view of conceptual systems, there are many heads that do not bear a heavy conceptual burden, such as “method” in English and many single-Chinese character elements in Japanese, which may be misleading in terms of the role of individual constituents vis-à-vis the conceptual system (these elements exist even after removing functional elements in constructing the networks).

Table 3 shows the number of terms, duplicated terms, constituents and duplicated constituents, and the distribution of subgraphs of the four types of terminological data compiled from the three terminologies. The data for DC is also indicated, to be referred to in section 5. In the subcolumns under ‘subgraphs’, ‘max’ means the largest subgraphs, and the values indicate the number of vertices in the largest subgraph. The columns ‘6’ to ‘1’ indicate the number of vertices in the subgraphs, and the values indicate the number of graphs with 6 vertices, 5 vertices, etc.

From Table 3 we can see that the graphs of all four domains are decomposed into a single very large graph and a number of subgraphs with at most six vertices. The task here is to partition the largest subgraphs in Table 3 into two for DC+AI, AI+GG and GG+DC and three for ALL, and evaluate to what extent the original terminologies of

different domains can be regrouped. Though the distributions of the subgraphs themselves reflect the motivated nature of the terminological structure, we will not deal with this aspect in this paper.

4.2 Results

Table 4 shows the results of the evaluation. Note that, except for GG+DC, there are duplicated terms, so the grand totals in Table 4 become the number of vertices in the largest subgraphs in Table 3 plus the duplicated terms in the largest subgraphs. Q indicates the modularity, and E_c indicates the number of removed edges.

Table 4. The results of clustering

(a) DC+AI ($Q = 0.4298$; $E_c = 2858$)

	DC	AI	Total
C1	403 (27.5)	1060 (72.5)	1463
C2	679 (59.5)	463 (40.5)	1142
Total	1082	1523	2605

(b) AI+GG ($Q = 0.4587$; $E_c = 1444$)

	AI	GG	Total
C1	119 (8.8)	1232 (91.2)	1351
C2	1386 (91.8)	123 (8.2)	1509
Total	1505	1355	2860

(c) GG+DC ($Q = 0.4662$; $E_c = 952$)

	GG	DC	Total
C1	1157 (98.1)	22 (1.9)	1179
C2	195 (15.9)	1031 (84.1)	1226
Total	1352	1053	2405

(d) ALL ($Q = 0.5898$; $E_c = 5057$)

	DC	AI	GG	Total
C1	398 (33.1)	690 (57.5)	113 (9.4)	1201
C2	567 (41.8)	737 (54.4)	51 (3.8)	1355
C3	121 (8.5)	106 (7.4)	1196 (84.0)	1423
Total	1086	1533	1360	3979

Though the modularities (Q) are not high, the terminologies of distant domains are separated well. Although there are no exactly comparable studies, and the simple quantitative performance of the partition itself is not the main concern of this paper, the performance of groupings for AI+GG, GG+DC and ALL in relation to GG, which range from around 84% to 98%, are comparable to the results of many experiments in text or word clustering and classification. The terminologies of closer domains are, on the other hand, mixed, as expected. This indicates: (i) the terminological networks constructed through common constituent elements contain, as a mass, the structural information of terminologies, as we discussed in section 2 to a substantial degree; and (ii) the Potts model-based clustering, which gives a straightforward terminological interpretation, provides a useful means for exploring the terminological networks.

Table 5. Basic quantities of the maximum-modularity partitions

	# groups	Q	E_c	Group size mean	max	min
DC	24	0.8357	1823	43.6	109	4
DC+AI	31	0.7726	26837	82.9	220	5

5 Qualitative observations

5.1 Experimental setup

For the qualitative observation of the terminological network, we used the DC terminology, which is familiar to the authors. We also used DC+AI terminologies, for the purpose of comparison. We selected DC+AI because, as shown below, mixed terminologies of AI+GG and GG+DC are rather clearly partitioned at the topmost level. This implies that the subsequent partitioning will be carried out mostly within the terminologies of the original domain. We wanted a terminology of an 'intermingled' nature, in order to guide the qualitative characterisation of the DC terminology.

The DC row in Table 2, and the DC and DC+AI rows in Table 3, show the basic quantities of the data. For these data, we applied the method by changing the parameter of the maximum number of clusters, which corresponds to the maximum number of possible spin states. The clustering results with maximum modularity value were chosen for the qualitative observations.

The essential point here is twofold: (i) if a given terminology is conceptually more dense and coherent, and linguistically represented more systematically with respect to the conceptual structure of the domain, the result of the clustering or detected communities should reflect both its coherency at the level of concepts and its systematicity at the level of linguistic representations, and therefore should be interpreted as meaningful; and (ii) correspondingly, if a given terminology is systematic, the ground state reached by the Potts model should show a lower Hamiltonian value, which means a higher modularity.

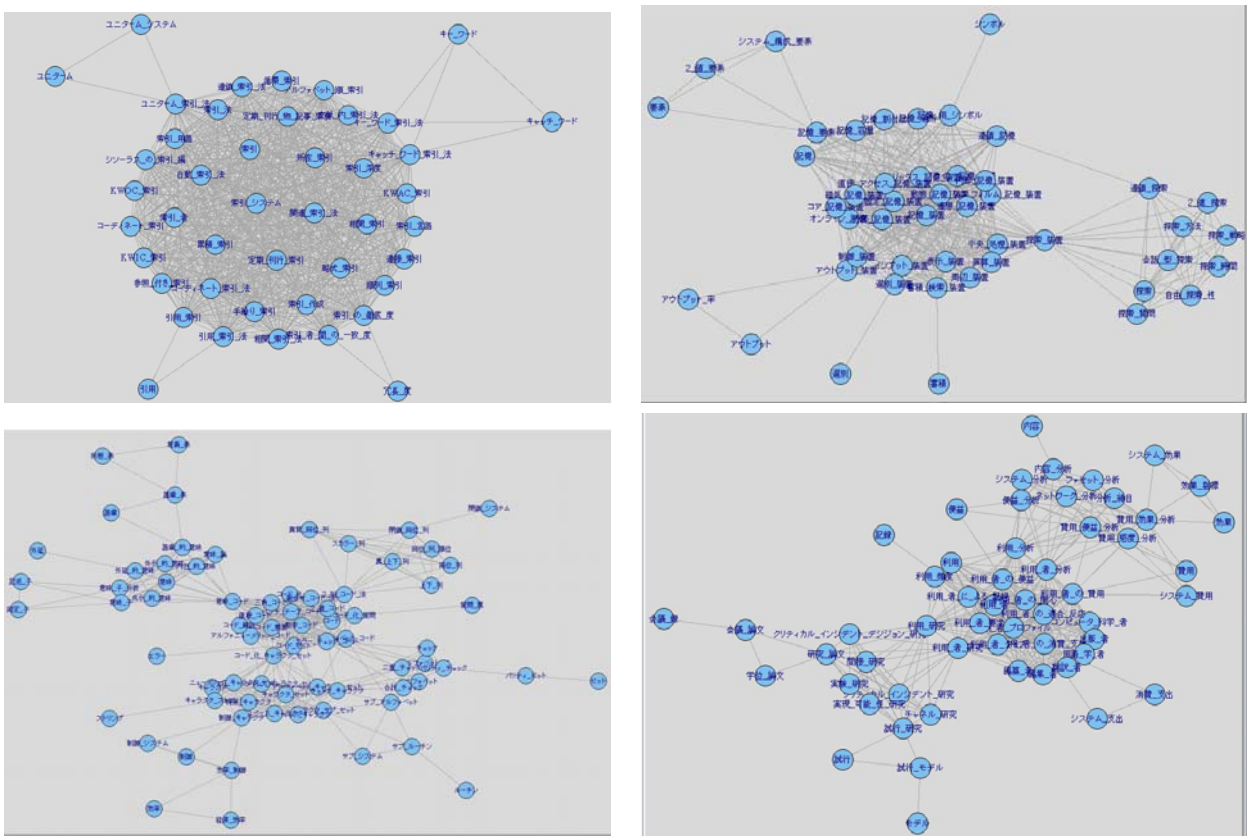


Figure 1. A network constructed from the sample terminology in Table 1. Top left: A (“indexing”); top right: B (“storage and retrieval”); bottom left: C (“bits, codes, characters and alphabets”); bottom right: D (“analysis, research, researchers, users”).

5.2 General tendencies

Table 5 shows the quantitative data for the clustering results with maximum modularities for the DC and DC+AI terminologies. The number of communities for DC is 24 and that for DC+AI is 31. Incidentally, for DC+AI, 12 communities consist of more than 80% AI terms, and 6 consist of more than 80% DC terms. The other 13 communities are highly hybrid. It is notable that, for DC, the modularity attains a high value of 0.8357 with the removal of less than 2,000 edges, while for DC+AI, the maximum modularity is attained by the removal of 26,837, nearly 15 times the number of removed edges for DC. The modularity value 0.7726 of DC+AI is still much lower than that of DC. These reflect the difference between the coherent terminology of DC and the hybrid set of terms consisting of DC+AI.

We visualised the in-community network using the Kamada-Kawai algorithm (Kamada and Kawai, 1989) in order to analyse the resultant communities. The following four types of community were identified for both DC and DC+AI terminologies:

A. More or less coherent cluster with a single center and possibly with some peripherals (less than 20%), to which we can assign a single generic label relevant to the domain;

B. More or less coherent clusters with more than one center and possibly with some peripherals, to which we can assign a generic label relevant to the domain;

C. Clusters with more than one center and some peripherals, to which we can assign a set of related labels relevant to the domain;

D. Clusters with multiple centers and peripherals, to which we can only assign either a set of unrelated labels or a very general, non-domain specific label.

An example of each type is shown in Figure 2.

Table 6 shows the classification of the communities according to A–D. For intuitive understanding we assigned labels to the communities. For DC+AI, we also added the label AI or DC for those which consist of mostly (more than 80%) AI terms or DC terms.

Table 6. The classification of communities detected in DC and AI+DC

	DC	DC+AI
A.	15 communities: "cataloguing", "communication", "library", "bibliography", "indexing", "monographs", "relations", "features", "prints", "classification", "abstracts", "words", "data", "information", "access"	13 communities: "model" (AI), "words", "library" (DC), "transformation" (AI), "classification and clustering", "indexing" (DC), "conditions" (AI), "relation", "knowledge" (AI), "bibliography" (DC), "copying" (DC), "symbolic representations" (DC), "communication" (DC)
B.	3 communities: "storage and retrieval", "copies and prints", "symbols and symbolic representations"	3 communities: "data and data processing", "pattern and recognition" (AI), "storage and multiplication devices and methods"
C.	5 communities: "bits, codes, characters and alphabets", "logic and diagrams", "languages and programs", "headings and entries", "cards, tapes and codings"	6 communities: "bits, codes, characters and alphabets", "logic, computation, signs", "graphs and reductions" (AI), "grammar and concepts" (AI), "information and related concepts", "learning and search" (AI)
D.	1 community: "analysis, research, researchers, users"	9 communities: "CAI, expert system, evaluation" (AI), "space, problems" (AI), "language, programs, concepts, robots", "catalogue, cards, copies", "reasoning, levels, representations" (AI), "terms, entries, monographs, retrieval" (DC), "automatic, misc.", "features, sensors", "semantics and discourse, misc." (AI)

5.3 Interpretations and diagnosis

The modularity values in Table 5 and the distribution of the types of community in Table 6 clearly show that the DC terminology has a much more motivated structure, which shows the dual systematicity of both the underlying concept system and also the way terms are materialised within the terminological space. A large number of type D communities for the DC+AI terminology can be interpreted as the mutual interference between terms that follow the AI system and those that follow the DC system.

To examine this further, we looked at the structure of the network within the communities. The major causes of 'hybridness' are as follows:

1. The existence of a few terms consisting of two or more highly meaningful and frequent elements. A typical example is "program language". This term attracts all the terms containing "program" as a constituent as well as all the terms containing "language". We thus have a hybrid community of "language, programs, concepts, robots" in the DC+AI terminology. In this case, even though terms belonging to different groups are classified in the same community, they tend to make chunks in the community. If the two constituents are conceptually close to each other, the resultant community will become type B (the group "symbols and symbolic representations" from DC is constructed by the existence of "representation symbol"). The latter case is expected to be the standard case for a systematic terminology of a single domain. This exists in the DC terminology as well as in the DC+AI terminology.

2. The existence of constituent elements which occur frequently but represent only secondary concepts. Examples are "automatic", "method", etc. This leads to a community such as "automatic, misc.", in which we cannot identify conceptual consistencies. An interesting point is that for the DC terminology, this kind of element does not contribute to making hybrid communities. The elements with heavier conceptual content might have suppressed the effect of these elements in a systematic terminology. Though observations of a wider range of terminologies are necessary, it might well be the case that this sort of community is unlikely to be formed for a coherent terminology representing a coherent domain. Incidentally, for the analysis of terminological structures, it would be interesting to analyse the community structure exploratively by adding or subtracting these constituents. For the practical aim of obtain-

ing good clustering results — though it is not our main concern here — excluding these constituents by such measures as *tfidf* in the construction of the network might well give better results.

3. Hybrid communities the causes of which are not clearly identifiable. A typical example is the community "analysis, research, researchers, users". It seems natural to regard this type of hybrid community as a "dust bin", in which a small number of peripheral terms are chunked together through the community detection process. The number of this kind of community is low in DC (only one) but three in DC+AI ("space, problems", "semantics and discourse, misc.", "automatic, misc.").

Going back to Table 6, it is interesting and important to observe the relationships between communities. For instance, although a substantial ratio of communities have high internal coherence in the DC terminology in its minimum modularity ground state, how larger communities are formed also matters for exploring the motivated structure of terminologies. It would be helpful to use the method hierarchically, restricting the number of communities that can be formed. We will cover this in the next step of our research.

6 Conclusion

Starting from defining the terminological networks constructed by shared constituent elements, we have (i) clarified the theoretical status of these networks for the study of terminology, (ii) adopted the Potts spin glass model-based community detection method to explore the networks and argued that the method allows a very natural interpretation of the conceptual characteristics of terminological structures and networks which is theoretically relevant; (iii) quantitatively evaluated the performance of the method from the point of view of clustering; and (iv) examined the motivated nature of the terminology of documentation, contrasting it with the combined terminologies of artificial intelligence and documentation.

We have shown that, through a relevant definition of terminological structure and a suitable method, it is possible to explore the terminological space bound by the conceptual system and represented by a set of terms, whose characteristics are elegantly postulated by Rey (1995):

> To the extent that a terminological system, even if it matches a coherent conceptual system, is incapable of reflecting its internal relationships, terminology is autonomous with respect to epistemology. To the extent that a terminological system, even if it is formed from unmotivated and common language words, denotes a conceptual system and exists only for denoting it, terminology is autonomous with respect to linguistics.

The Potts spin glass-based model provides us with a means to explore this terminological structure in such a way that we can interpret the results in terms of the motivatedness of the conceptual-linguistic systems. We are currently carrying out in-depth analyses of the motivated structure of terminologies of several domains, with the conceptual nature and status of constituent elements also under consideration.

So far we have deliberately distanced ourselves from placing this work within the framework of clustering applications, for reasons which include that already discussed in section 3.2. This does not mean that what we have introduced in this paper is not relevant for practical applications. One possible application is in the selection of headwords for terminological lexicons. Of particular importance in this process is maintaining the coherency of the selected headwords as a set. This process depends to a great extent on the sense of the lexicographer, and little formalisation has been carried out so far. No automatic methods have been successful, either. Given this situation, it would be useful if the editors could check the structure and consistency of headwords *en masse* through explorative means. What we have discussed in this paper could potentially contribute in this regard.

References

Aizawa, A. and Kageura, K. 1998. An approach to the automatic generation of multilingual keyword clusters. *Computerm'98*, 8–14.

Albert, R. and Barabasi, A-L. 2002. Statistical mechanics of complex networks. *Review of Modern Physics*, 74:47–97.

Cheng, C-K. and Wei, Y-C. A. 1991. An improved two-way partitioning algorithm with stable performance. *IEEE Transactions on Computer-Aided Design*, 10(12):1502–1513.

Csardi, G. 2006. The igraph library. http://cneurocvs.rmki.kfki.hu/igraph/

Daille, B. et. al. 1996. Empirical observation of term variations and principles for their description. *Terminology*, 3(2):197–258.

Danon, L. et. al. 2005. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, P09008.

Dorogovtsev, S. N. et. al. 2004. Potts model on complex networks. *European Physical Journal*, 38:177–182.

Duch, J. and Arenas, A. 2005. Community detection in complex networks using extremal optimization. *Physical Review E*, 72:027104.

Fortunato, S. et. al. 2004. Method to find community structures based on information centrality. *Physical Review E*, 70:056104.

Fujishige, S. 2002. *Graph, Network and Combinatorics*. Tokyo: Kyoritsu.

Hamon, T. and Nazarenko, A. 2001. Detection of synonymy links between terms. Bourigault, D. et. al. eds. *Recent Advances in Computational Terminology*. Amsterdam: John Benjamins. 185–208.

Handl, J., et. al. 2003. On the performance of ant-based clustering. *Frontiers in Artificial Intelligence and Applications*, 104:204–213.

Heid, U. 1999. A linguistic bootstrapping approach to the extraction of term candidates from German text. *Terminology*, 5(2):161–182.

Jacquemin, C. 2001. *Spotting and Discovering Terms through Natural Language Processing*. Cambridge, Mass: MIT Press.

Jain, A. K. et. al. 1999. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

JME: Japanese Ministry of Education, ed. 1981. *Japanese scientific terms: geography*. Tokyo: JSPS.

JSAI: Japan Society of Artificial Intelligence. 1990. *AI Handbook*. Tokyo: Ohm.

Kageura, K. 1998. A preliminary investigation of the nature of frequency distributions of constituent elements of terms in terminology. *Terminology*, 4(2):199-223.

Kageura, K. 2002. *The Dynamics of Terminology*. Amsterdam: John Benjamins.

Kamada, T. and Kawai, S. 1989. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15.

Masuda, N. and Konno, N. 2005. *Introduction to Complex Networks*. Tokyo: Sangyotosyo.

Matsuo, Y. et. al. 2006. Graph-based word clustering using web search engine. *EMNLP 2006*.

Newman, M. E. J. 2003. The structure and function of complex networks. *SIAM Review*, 45(2):167–256.

Newman, M. E. J. 2004. Detecting community structure in networks. *European Physical Journal*, 38:321–330.

Newman, M. E. J. and Girvan, M. 2004. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113.

Nishimori, H. 1999. *Spin Glass Theory and Statistical Mechanics of Information*. Tokyo: Iwanami.

Nomura, M. and Ishii, M. 1988. *List of Stems in Japanese Technical Terms*. Tokyo: National Language Research Institute.

Reichardt, J. and Bornholdt, S. 2006. Statistical mechanics of community detection. *Physical Review E*, 74:016110.

Rey, A. 1995. *Essays on Terminology*. Amsterdam: John Benjamins.

Sager, J. C. 1991. *A Practical Course in Terminology Processing*. Amsterdam: John Benjamins.

Saussure, F. 1910-11. *Troisième Cours de Linguistique Générale — noté par Émile Constantin* (Msfr 3972-3). Geneva: Bibliothèque publique et universitaire.

Scott, J. 2000. *Social Network Analysis: A Handbook*. London: Sage.

Strogatz, S. H. 2001. Exploring complex networks. *Nature*, 410:268–276.

Takamura, H. et. al. 2005. Extracting semantic orientation of words using spin model. *ACL 2005*, 133–140.

Watts, D. J. and Strogatz, S. H. 1998. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442.

Weeds, J. et. al. 2005. Using distributional similarity to organise biomedical terminology. *Terminology*, 11(1):107–141.

Wersig, G. and Neveling, U. eds. 1986. *UNESCO Documentation Terms*. Paris/Tokyo: UNESCO/JICST.

Grabar, N. and Zweigenbaum, P. 2004. Lexically-based terminology structuring. *Terminology*, 10(1):23–53.

Determining the Unithood of Word Sequences using Mutual Information and Independence Measure

Wilson Wong, Wei Liu and Mohammed Bennamoun
School of Computer Science and Software Engineering
University of Western Australia
Crawley WA 6009
{wilson,wei,bennamou}@csse.uwa.edu.au

Abstract

Most works related to unithood were conducted as part of a larger effort for the determination of termhood. Consequently, the number of independent research that study the notion of unithood and produce dedicated techniques for measuring unithood is extremely small. We propose a new approach, independent of any influences of termhood, that provides dedicated measures to gather linguistic evidence from parsed text and statistical evidence from Google search engine for the measurement of unithood. Our evaluations revealed a precision and recall of 98.68% and 91.82% respectively with an accuracy at 95.42% in measuring the unithood of 1005 test cases.

1 Introduction

Terms and the tasks related to their treatments are an integral part of many applications that deal with natural language text such as large-scale search engines, automatic thesaurus construction, machine translation and ontology learning for purposes ranging from indexing to cluster analysis. With the increasing reliance on huge text sources such as the World Wide Web as input, the need to provide automated means for managing domain-specific terms rises. Such relevance and importance of terms has prompted dedicated research interests. Various names such as *automatic term recognition*, *term extraction* and *terminology mining* were given to encompass the tasks related to the treatment of terms. Term extraction is the process of extracting lexical units from text and filtering them for the purpose of identifying terms which characterise certain domains of interest. This process is the determination of two important factors, namely, *unithood* and *termhood*. Unithood concerns with whether or not a sequence of words should be combined to form a more stable lexical unit, and termhood measures the degree to which these stable lexical units are related to domain-specific concepts. While unithood is only relevant to *complex terms* (i.e. multi-word terms), termhood concerns both *simple terms* (i.e. single-word terms) and complex terms.

Most research in automatic term recognition were conducted solely to study and develop techniques for measuring termhood, while only a small number exists that study on unithood. Unfortunately, rather than considering the measurement of unithood as an important prerequisite, these researchers merely treat it as part of a larger scoring and filtering mechanism for determining termhood. Such perception is clearly reflected through the words of Kit (2002), "*...we can see that the unithood is actually subsumed, in general, by the termhood.*". Consequently, the significance of unithood measures has been overshadowed by the larger notion of termhood. As such, the progress and innovation with respect to this small sub-field of automatic term recognition is minimal. Most of the existing techniques for measuring unithood employ conventional measures such as mutual information and log-likelihood, and rely simply on the occurrence and co-occurrence frequencies from domain corpora as the source of evidence.

In this paper, we propose the separation of unithood measurements from the determination of termhood. From here on, we will consider unithood measurement as an important prerequisite, rather than a subsumption, to the determination of termhood. We present a new dedicated approach for determining the unithood of word sequences by employing the Google search engine as the source of statistical evidence, and measures inspired by mutual information (Church and Hanks (1990)) and *Cvalue* (Frantzi (1997)). The use of the World Wide Web to replace the conven-

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 246–254,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

tional use of static corpora will eliminate issues related to portability to other domains, and the size of text necessary to induce the required statistical evidence. Besides, this dedicated approach to determine the unithood of word sequences will prove to be invaluable to other areas in natural language processing such as noun-phrase chunking and named-entity recognition. Our evaluations revealed a precision and a recall of 98.68% and 91.82% respectively with an accuracy at 95.42% in measuring the unithood of 1005 test cases.

In Section 2, we have a brief review on the existing techniques for measuring unithood. In Section 3, we present our new approach, the measures involved and the justification behind every aspect of the measures. In Section 4, we summarize some findings from our evaluations. We discuss in Section 5 why our new approach can be applicable to other tasks in natural language processing such as named-entity recognition. Finally, we conclude this paper with an outlook to future works in Section 6.

2 Related Works

Prior to measuring unithood, term candidates must be extracted. There are two common approaches for extracting the term candidates. The first requires the corpus to be tagged or parsed, and a filter is then employed to extract words or phrases satisfying some linguistic patterns. There are two types of filters for extracting from tagged corpus, namely, open or closed. Too restricted filters (i.e. closed) that rely on a small set of allowable part-of-speech will produce high precision but poor recall (Frantzi and Ananiadou (1997)). On the other hand, filters that are too liberal (i.e. open), allowing part-of-speech such as prepositions and adjectives, will have the opposite effect. Most of the existing approaches rely on regular expressions and the part-of-speech tags to accept or reject sequences of n-grams as term candidates. For example, Frantzi and Ananiadou (1997) employ Brill tagger to tag the raw corpus with part-of-speech and later extract n-grams that fulfill the pattern $(Noun|Adjective)^{+}Noun$. Bourigault and Jacquemin (1999) utilise SYLEX, a part-of-speech tagger, to tag the raw corpus. The part-of-speech tags are utilised to extract maximal-length noun phrases, which are later recursively decomposed into heads and modifiers. On the other extreme, Dagan and Church (1994) accept only sequences of $Noun+$. The second type of extraction approaches works on raw corpus using a set of heuristics. This type of approaches which does not rely on part-of-speech tags is quite rare. Such approaches have to make use of the textual surface constraints to approximate the boundaries of term candidates. One of the constraints include the use of a stopword list to obtain the boundaries of stopwords for inferring the boundaries of candidates. A selection list of allowable prepositions can also be employed to enforce constraints on the tokens between units.

The filters for extracting term candidates make use of only local, surface-level information, namely, the part-of-speech tags. More evidence is required to establish the dependence between the constituents of each term candidate to ensure strong unithood. Such evidence will usually be statistical in nature in the form of co-occurrences of the constituents in the corpus. Accordingly, the unithood of the term candidates can be determined either as a separate step or may proceed as part of the extraction process. From our review of the literature, only an extremely small number of researchers actually discussed and presented measures for unithood. The lack of extensive research and techniques related specifically to unithood is reaffirmed by Kit (2002). According to the author, *"...its measure (if there is one) indicates how likely it is that a term candidate is an atomic text unit."*

Two of the most common measures of unithood have to be pointwise *mutual information (MI)* (Church and Hanks (1990)) and *log-likelihood ratio* (Dunning (1994)). In mutual information, the co-occurrence frequencies of the constituents of complex terms are utilised to measure their dependency. The mutual information for two words a and b is defined as:

$$MI(a,b) = \log_2 \frac{p(a,b)}{p(a)p(b)} \quad (1)$$

where $p(a)$ and $p(b)$ are the probabilities of occurrence of a and b. Many measures that apply statistical techniques assuming strict normal distribution, and independence between the word occurrences do not fare well. For handling extremely uncommon words or small sized corpus, *log-likelihood ratio* delivers the best precision (Kurz and Xu (2002); Franz (1997)). Log-likelihood ratio attempts to quantify how much more likely one pair of words is to occur compared to the others. Despite its potential, *"How to apply*

this statistic measure to quantify structural dependency of a word sequence remains an interesting issue to explore." (Kit (2002)).

Frantzi (1997) proposed a measure known as *Cvalue* for extracting complex terms. The measure is based upon the claim that a substring of a term candidate is a candidate itself given that it demonstrates adequate independence from the longer version it appears in. For example, *"E. coli food poisoning"*, *"E. coli"* and *"food poisoning"* are acceptable as valid complex term candidates. However, *"E. coli food"* is not. Therefore, some measures are required to gauge the strength of word combinations to decide whether two word sequences should be merged or not. Given a word sequence a to be examined for unithood, the *Cvalue* is defined as:

$$Cvalue(a) = \begin{cases} \log_2 |a| \cdot f_a & \text{if } |a| = g \\ \log_2 |a| \cdot (f_a - \frac{\sum_{l \in L_a} f_l}{|L_a|}) & \text{otherwise} \end{cases} \quad (2)$$

where $|a|$ is the number of words in a, L_a is the set of longer term candidates that contain a, g is the longest n-gram considered, f_a is the frequency of occurrence of a, and $a \notin L_a$. While certain researchers (Kit (2002)) consider *Cvalue* as a termhood measure, others (Nakagawa and Mori (2002)) accept it as a measure for unithood. One can observe that longer candidates tend to gain higher weights due to the inclusion of $log_2|a|$ in Equation 2. In addition, the weights computed using Equation 2 are purely dependent on the frequency of a.

3 A New Approach for Unithood Measurement

Our new approach for measuring the unithood of word sequences consists of two parts. Firstly, a list of word sequences is extracted using purely linguistic techniques. Secondly, word sequences are examined and the related statistical evidence is gathered to assist in determining their mutual information and independence.

3.1 Extracting Word Sequences

Existing techniques for extracting word sequences have been relying on part-of-speech information and filters in the form of pattern matching (e.g. regular expression). Since the head-modifiers principle is important for our techniques, we employ both the part-of-speech information and dependency relation for extracting term candidates. The filter is implemented as a head-driven left-right filter (Wong (2005)) that feeds on the output of Stanford Parser (Klein and Manning (2003)), which is an implementation of unlexicalised *probabilistic context-free grammar (PCFG)* and lexical dependency parser. The head-driven filter begins by identifying a list of head nouns from the output of the Stanford Parser. As the name suggests, the filter begins from the head and proceeds to the left and later, right in the attempt to identify maximal-length noun phrases according to the head-modifier information. During the process, the filter will append or prepend any immediate modifier of the current head which is a noun (except possessive nouns), an adjective or a foreign word. Each noun phrase or segment of noun phrase identified using the head-driven filter is known as a *potential term candidate*, $a_i \in A$ where i is the word offset produced by the Stanford Parser (i.e. the *"offset"* column in Figure 1).

Figure 1 shows the output of the Stanford Parser for the sentence *"They're living longer with HIV in the brain, explains Kathy Kopnisky of the NIH's National Institute of Mental Health, which is spending about millions investigating neuroAIDS."*. Note that the words are lemmatised to obtain the root form. The head nouns are marked with squares in the figure. For example, the head *"Institute"* is modified by *"NIH's"*, *"National"* and *"of"*. Since we do not allow for modifiers of the type *"possessive"* and *"preposition"*, we will obtain *"National Institute"* as shown in Figure 2. Figure 2 shows the head-driven filter at work for some of the head nouns identified from the *"modifiee"* column of the output in Figure 1. After the head-driven filter has identified potential term candidates using the heads, remaining nouns from the *"word"* column in Figure 1 which are not part of any potential term candidates will be included in A.

3.2 Determining the Unithood of Word Sequences

In the following step, we examine the unithood of all pairs of potential term candidates $(a_x, a_y) \in A$ with a_x and a_y located immediately next to each other (i.e. $x+1=y$), or separated by a preposition or coordinating conjunction *"and"* (i.e. $x+2=y$). Obviously, a_x has to appear before a_y in the sentence or in other words, $x < y$ for all pairs where x and y are the word offsets produced by the Stan-

offset	word	partofspeech	dependency	modifiee	modifiee offset
2	They	PRP	nsubj	live	4
3	are	VBP	aux	live	4
4	live	VBG	ccomp	explains	13
6	with	IN	dep	long	5
7	HIV	NN	pobj	with	6
8	in	IN	prep	live	4
9	the	DT	det	brain	10
10	brain	NN	pobj	in	8
14	Kathy	NNP	nn	Kopnisky	15
15	Kopnisky	NNP	nsubj	explains	13
16	of	IN	prep	Kopnisky	15
17	the	DT	det	NIH's	18
18	NIH's	NNP	poss	Institute	21
20	National	NNP	nn	Institute	21
21	Institute	NNP	pobj	of	16
22	of	IN	prep	Institute	21
23	Mental	NNP	nn	Health	24
24	Health	NNP	pobj	of	22
26	which	WDT	rel	spend	28
27	is	VBZ	aux	spend	28
28	spend	VBG	rcmod	Kopnisky	15
30	million	CD	dobj	spend	28
31	investigate	VBG	partmod	spend	28
32	neuroAIDS	NNS	dobj	investigate	31

Figure 1: The output from Stanford Parser. The tokens in the *"modifiee"* column marked with squares are head nouns, and the corresponding tokens along the same rows in the *"word"* column are the modifiers. The first column *"offset"* is subsequently represented using the variable i.

ford Parser. Formally, given that $s = a_x b a_y$ where b is any preposition, the conjunction *"and"* or an empty string, the problem is to determine whether to accept s as an independent lexical unit (i.e. a term candidate) or leave a_x and a_y as separate units. In order to decide on the merge, we need adequate evidence that s will form a stable unit and hence, a better term candidate than a_x and a_y separated. It is worth mentioning that the size (i.e. number of words) of a_x and a_y is not limited to 1. For example, we can have a_x=*"National Institutes"*, b=*"of"* and a_y=*"Allergy and Infectious Diseases"*. In addition, the size of a_x and a_y should have no effect on the determination of their unithood.

The most suitable existing measure for gathering evidence about the dependency between two words is mutual information. Based on the conventional practice, the frequency of occurrence of each element in $W = \{s, a_x, a_y\}$ is normalized using the sum of the frequency of all occurrences in W. Since data sparseness in the local corpus may lead to poor estimation of mutual information, we innovatively employ the page count by Google search engine for calculating the dependency of the elements in W instead. We treat the World Wide Web as a large general corpus and Google search engine as a gateway for accessing the documents in the corpus. Our choice of using Google to obtain the page count was merely motivated by its extensive coverage. In fact, it is possible to employ any search engines on the World Wide Web for this research. Each element in W is formulated as a search query and submitted to Google search engine. The page count returned is utilised for calculating the mutual information. In addition, we also apply a multiplier within the range of $[e^{-1}, 1]$, inspired by TF-IDF, to offset too common terms, especially in three-word terms such as *"Institute of Science"*. Formally, for each $w \in W$, we define the weight as:

$$p(w) = \frac{n_w}{\sum_{v \in W} n_v} e^{\left(-\frac{n_w}{\sum_{v \in W} n_v}\right)} \tag{3}$$

where n_w is the page count (i.e. number of documents) returned by Google search engine containing $w \in W$. We only take into consideration the number of documents that contain the word sequences (i.e. page count) due to the difficulty in obtaining the actual frequency of occurrences of word sequences from Google's search results. Next, the mutual information between the two units a_x and a_y is defined as:

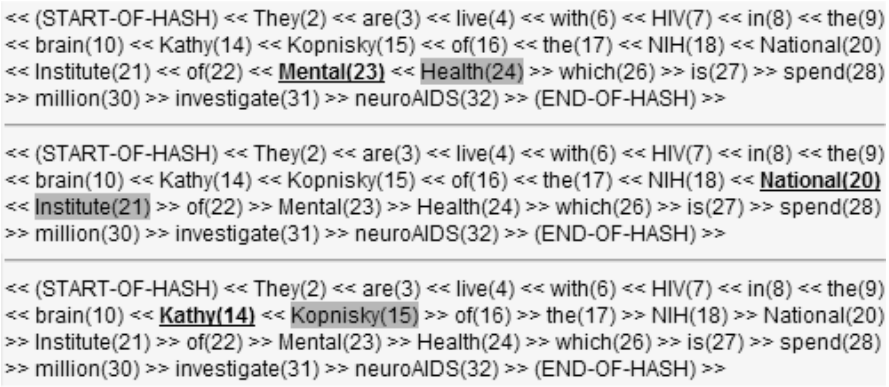


Figure 2: An example of our head-driven left-right filter at work. The tokens which are highlighted with a darker tone are the head nouns. The underlined tokens are the modifiers identified as a result of the left-first and right-later movement of the filter. In the first segment, the head noun is *"Health"* while the first token to the left *"Mental"* is the corresponding modifier. In the second segment of the figure, *"Institute"* is the head noun and *"National"* is the modifier. Note that the results from the first and second segments are actually part of a longer noun phrase. Due to the restriction in accepting prepositions by the head-driven filter, *"of"* is omitted from the output of the first segment of the figure.

$$MI(a_x, a_y) = \frac{p(s)}{p(a_x)p(a_y)} \tag{4}$$

If the occurrence of s approaches 0 (i.e. a_x and a_y are rarely seen together as a unit with b), then $MI(a_x, a_y)$ reduces to 0. If the occurrence of s approaches that of the co-occurrence of a_x and a_y, $MI(a_x, a_y)$ will approach 1. A high $p(s)$ indicates stronger coupling of a_x and a_y, or even more so, the existence of a_x and a_y are purely due to the existence of s. A high $MI(a_x, a_y)$ implies an increase in the unithood of the two units a_x and a_y. Following this, a_x and a_y will have poor unithood in their individual forms if they are not merged into s to form a stronger unit. This mutual information measure is necessary in distinguishing phrases such as *"Asia and Europe"* (i.e. low mutual information) from *"U.S. Food and Drug Administration"* (i.e. high mutual information) when prepositions and conjunctions are involved.

Nonetheless, the units a_x and a_y may still be capable of forming valid compound unit s even though their mutual information is relatively low. Low mutual information can be attributed to the high individual occurrences of a_x and a_y due to their extremely common usage. For example, a_x=*"Institute"* and a_y=*"Ophthalmology"* yield high occurrences relative to s=*"Institute of Ophthalmology"*. This does not mean that *"Institute of Ophthalmology"* is not a valid unit. To handle such cases where $MI(a_x, a_y)$ is mediocre due to the commonness of a_x and a_y, we employ another measure of independence. In such situation, we will still accept s as a valid unit if it can be demonstrated that the extremely high independence of the individual unit a_x and a_y is the cause behind the low $MI(a_x, a_y)$. For this purpose, we modify the *Cvalue* described in Equation 2 to accommodate the use of page counts rather than frequency. In addition, we remove the multiplier $\log_2 |a|$ because the number of words in a_x and a_y does not play a role in determining their independence from s. Consequently, we define the measure of *Independence (ID)* for a_x and a_y from s as:

$$ID(a_x, s) = \begin{cases} \log_{10}(n_{a_x} - n_s) & \text{if}(n_{a_x} > n_s) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$ID(a_y, s) = \begin{cases} \log_{10}(n_{a_y} - n_s) & \text{if}(n_{a_y} > n_s) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where n_{a_x}, n_{a_y} and n_s is the Google page count for the unit a_x, a_y and s, respectively. As the lexical unit a_x occurs more than its longer counterpart s, its independence $ID(a_x, s)$ grows. Only when the number of occurrences of a_x is less than those of s, its independence from s becomes $ID(a_x, s) = 0$. This means that we will not be able to witness a_x without encountering s. The same can be said about the measure of independence for a_y, $ID(a_y, s)$. In short, extremely high independence of a_x and a_y relative to s will be reflected through high $ID(a_x, s)$ and $ID(a_y, s)$.

Consequently, the decision to merge a_x and a_y to form s depends on both the mutual information between a_x and a_y, namely, $MI(a_x, a_y)$, and the independence of a_x and a_y from s, namely, $ID(a_x, s)$ and $ID(a_y, s)$. This decision is organised into a Boolean function known as *Unithood* (UH), and we define it as:

$$UH(a_x, a_y) = \begin{cases} 1 & \text{if } (MI(a_x, a_y) > MI^+) \vee \\ & (MI^+ \geq MI(a_x, a_y) \\ & \geq MI^- \wedge \\ & ID(a_x, s) \geq ID_T \wedge \\ & ID(a_y, s) \geq ID_T \wedge \\ & IDR^+ \geq IDR(a_x, a_y) \\ & \geq IDR^-) \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where $IDR(a_x, a_y) = ID(a_x, s)/ID(a_y, s)$. IDR helps to ensure that pairs with mediocre mutual information not only have a_x and a_y with high independence but are also equally independent before mergings are performed. The unithood function in Equation 7 summarises the relationship between mutual information and the independence measure. $UH(a_x, a_y)$ simply states that the two lexical units a_x and a_y can only be merged in two cases:

- If a_x and a_y has extremely high mutual information (i.e. higher than a certain threshold MI^+); or
- If a_x and a_y achieve average mutual information (i.e. within the acceptable range of two thresholds MI^+ and MI^-) due to both of their extremely high independence (i.e. higher than the threshold ID_T) from s. To ensure that both units have equally high independence, their ratio of independence IDR has to fall within the range IDR^- and IDR^+.

The thresholds for $MI(a_x, a_y)$, $ID(a_x, s)$, $ID(a_x, s)$ and $IDR(a_x, a_y)$ are decided empirically through our evaluations:

- $MI^+ = 0.9$
- $MI^- = 0.02$
- $ID_T = 6$
- $IDR^+ = 1.35$
- $IDR^- = 0.93$

A general guideline for determining the appropriate threshold values is provided in the next section. Finally, the word sequence $s = a_x b a_y$ will be accepted as a stable lexical unit (i.e. term candidate) if and only if $UH(a_x, a_y) = 1$.

4 Evaluations and Discussions

Table 1: Contingency table constructed using the actual and ideal results for computing precision, recall, accuracy and F-score. *Ideal results* are used as a reference for evaluation. *Actual results* are the actual output from our new approach.

contingency table		ideal results		
		merged	not merged	
actual results	merged	449	6	455
	not merged	40	510	550
		489	516	1005

For this evaluation, we employ 300 news articles from Reuters in the health domain gathered between October 2006 to January 2007. These 300 articles are fed into the Stanford Parser whose output is then used by our head-driven left-right filter to extract word sequences in the form of nouns and noun phrases. Pairs of word sequences (i.e. a_x and a_y) located immediately next to each other, or separated by a preposition or the conjunction *"and"* in the same sentence are measured for their unithood. Based on the $UH(a_x, a_y)$ of the pairs, the decisions on whether to merge or not are done automatically. These decisions are known as the *actual results*. At the same time, we inspect the same list manually to decide on the merging of all the pairs. These decisions are known as the *ideal results*. Using the 300 news articles, we managed to obtain 1005 pairs of words to be tested for unithood. The actual and ideal results are organised into a contingency table as shown in Table 1 to identify the true and the false positives, and the true and the false negatives. Using the results in Table 1, we obtained a precision of 98.68%, a recall of 91.82% and an F-score of 90.61%. As for the accuracy, our new measures for unithood scored 95.42%. It shows that our new measure has very good precision and a relatively low recall due to the high number of false negatives.

Firstly, we realised that the high false negative rate is explained by our more conservative definition of the thresholds namely ID_T, IDR^+ and IDR^-. We discovered that about 90% of the false

no.	a_x	b	a_y	$ID(a_x,s)$	$ID(a_y,s)$	$IDR(a_x,a_y)$	$MI(a_x,a_y)$	$UH(a_x,a_y)$	s
66	Agriculture Department lab	in	Ames	1.6989	7.3765	0.2303	1.5466	MERGED	Agriculture Department lab in Ames
72	EU Commissioner	for	Health and Consumer Protection Markos Kyprianou	5.9029	2.7481	2.1479	0.7289	NOTMERGED	EU Commissioner for Health and Consumer Protection Markos Kyprianou
120	Atlanta-based Centers	for	Disease Control and Prevention	4.0583	5.9370	0.6835	0.1932	NOTMERGED	Atlanta-based Centers for Disease Control and Prevention
171	Joint Committee	on	Public Health	6.0488	8.2041	0.7372	0.0021	NOTMERGED	Joint Committee on Public Health
567	Center	for	Tobacco Research	8.9216	5.6280	1.5852	0.1529	NOTMERGED	Center for Tobacco Research

Figure 3: This figure shows a snapshot of some samples of false positives and false negatives taken from our evaluations. Row 66 is an example of merged pair which is not supposed to be combined (i.e. false positives) while the remaining rows are false negatives. Each row has two lexical units a_x (column 2) and a_y (column 4) to be examined for their mutual information (column 8), independence from s (column 5 and 6) to determine if they should be merged into s (column 10). The decision to merge (column 9) is accomplished based on Equation 7.

negatives fall within the range of MI^+ and MI^- (i.e. mediocre mutual information). Such pairs have the opportunity to be merged if they demonstrate adequate independence from s. Unfortunately, most of the independence ID of either one or both members of the pairs failed to satisfy our independence thresholds. For example, referring to rows 72, 120 and 567 as shown in Figure 3, one will notice that they have mediocre mutual information as defined by $MI^+ = 0.9$ and $MI^- = 0.02$. In the case of row 72, both a_x and a_y have an independence lower than the threshold $ID_T = 6$. This resulted in the decision of not merging them. The same case happened to row 120. In row 567, only a_y has an independence lower than ID_T and at the same time, their IDR is well above the upper limit IDR^+. The remaining 10% of the false negatives are simply due to the extremely low mutual information $MI(a_x, a_y)$ of the pairs. Take for example pair 171 in Figure 3 where the mutual information is only 0.002, which is way below MI^-. Secondly, due to the small number of false positives, not much conclusion can be drawn. From our analysis of the results, most of the false positives are due to their high mutual information (i.e. $MI(a_x, a_y)$ above MI^+). Pair 66 in Figure 3 is an example which is incorrectly merged due to a mutual information value higher than MI^+.

From our discussion above, one would realise that both the recall and the precision can be improved by adjusting the various thresholds. For most of the time, the improvement of one comes at the expense of the other. For example, we can improve the recall by lowering ID_T and broadening the range between IDR^+ and IDR^- at the expense of precision. In other words, more pairs with ID values exceeding the threshold ID_T and more pairs will fall within the range of acceptable IDR. In this case, we will lower the number of false negatives and hence, higher recall. Similarly, we can improve the precision by increasing MI^+. In this case, an increasing number of pairs will have mediocre mutual information (i.e. within the range MI^+ and MI^-). Consequently, the number of false positives will reduce when such pairs with mediocre mutual information are not merged due to their inability to satisfy additional constraints in the forms of ID_T, IDR^+ and IDR^-.

Due to the lack of existing dedicated techniques for measuring unithood, we were unable to perform a comparative study. Nonetheless, the high accuracy and F-score presented during our evaluation, and our analysis on the false positives and the false negatives revealed the potentials of our new measures in terms of high precision and recall, portability across domains, and configurability of the performance.

5 Applicability to Named-Entity Recognition

Named-entity recognition is one of the important tasks in information extraction. It involves the identification of noun phrases or more specifically, proper names from free text, and their classification into one of the many categories such as persons, geographical locations and companies. A typical named-entity recogniser performs part-of-speech tagging, and rely on patterns, heuristics and dictionary to identify proper names. The use of machine learning and other probabilistic methods such as support vector machines (Mayfield et al. (2003)) and hidden Markov models (Mittal et al. (1999)) have also gained popularity.

Most of these existing techniques for named-entity recognition works well when they are dealing with single-word names or sequences of nouns. In the face of more complex named-entities that consist of other part-of-speech especially prepositions, these techniques performed poorly. For example, most existing techniques would have to rely on heuristics and dictionaries to differentiate between *"Barrow in Furness"* and *"countries in Asia"*. Unfortunately, there are many problems related to the use of heuristics and dictionaries. For one, the maintenance of such dictionaries are costly and difficult. Questions about the reliability of stop-word boundaries, words capitalisation and punctuations arise regarding the use of heuristics and linguistics cues. For example, how can named-entity recognisers rely on capitalised words in the case of *"Barrow in Furness"* versus *"Perth in Australia"*? Besides prepositions, the conjunction *"and"* in named-entities posed similar challenge. According to Osenova and Kolkovska (2002), *"...problem arises when named-entity is a phrase, comprising a conjunction..."*. In such cases, the named-entity recogniser only recognises part of the named-entity. For example, in the case of *"Centre for Disease Control and Prevention"*, only *"Centre for Disease Control"* will be extracted. Other researchers (Mani et al. (1996)) have taken the default step of simply grouping name segments separated by prepositions or conjunctions into longer names.

Our new approach of deciding on whether two word sequences are to be merged or not is highly applicable to many areas in natural language processing especially named-entity recognition. The absence of any predefined resources in our approach will solve all the problems highlighted in the previous paragraph. Using our $UH(a_x, a_y)$ function, named-entity recogniser can easily determine whether or not parts of proper names should be merged together without ever relying on unreliable heuristics, and domain-restricted patterns and dictionaries.

6 Conclusion and Future Work

Many researchers inappropriately assume that termhood subsumes unithood. In this paper, we highlighted the significance of unithood and that its measurement should be given equal attention by researchers in automatic term recognition. The potential of unithood measurements can be extended to other areas in natural language processing such as noun-phrase chunking and named-entity recognition.

We proposed a new approach that provides dedicated measures specialised in measuring unithood. The first measure employs mutual information $MI(a_x, a_y)$ to capture the interdependence of the existence of a_x and a_y, and allows us to determine if a_x and a_y are better off separated or otherwise in order to produce a stronger unit. The second measure, defined as *Independence (ID)* was inspired by *Cvalue*, and is meant to provide additional evidence in the determination of unithood. These two measures are combined into a Boolean function defined as *Unithood* (UH) that decides on whether a_x and a_y should be combined to form s.

Our evaluations revealed a precision and recall of 98.68% and 91.82% respectively with an accuracy at 95.42% in measuring the unithood of 1005 test cases. Due to the lack of existing dedicated techniques for measuring unithood, we were unable to perform a comparative study. Nonetheless, the excellent evaluation results together with the real-world text employed in our evaluation demonstrate the strengths of our new approach in regards to the determination of unithood. One of the future works that we plan to undertake is to increase the size of our test set to further establish the advantages of our new approach demonstrated through the current evaluations. We are planning to reduce the number of thresholds and at the same time, find ways to automatically optimise them.

Acknowledgement

This research was supported by the Australian Endeavour International Postgraduate Research Scholarship, and the Research Grant 2006 by the University of Western Australia.

References

D. Bourigault and C. Jacquemin. 1999. Term extraction + term clustering: An integrated platform for computer-aided terminology. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*. Bergen.

K. Church and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

I. Dagan and K. Church. 1994. Termight: Identifying and translating technical terminology. In *Proceedings of the 4th Conference on Applied Natural Language Processing*. Germany.

T. Dunning. 1994. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

K. Frantzi. 1997. Incorporating context information for the extraction of terms. In *Proceedings of the 35th Annual Meeting on Association for Computational Linguistics*. Spain.

K. Frantzi and S. Ananiadou. 1997. Automatic term recognition using contextual cues. In *Proceedings of the IJCAI Workshop on Multilinguality in Software Industry: the AI Contribution*. Japan.

A. Franz. 1997. Independence assumptions considered harmful. In *Proceedings of the 8th Conference on European Chapter of the Association for Computational Linguistics*. Madrid, Spain.

C. Kit. 2002. Corpus tools for retrieving and deriving termhood evidence. In *Proceedings of the 5th East Asia Forum of Terminology*. Haikou, China.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

D. Kurz and F. Xu. 2002. Text mining for the extraction of domain relevant terms and term collocations. In *Proceedings of the International Workshop on Computational Approaches to Collocations*. Vienna.

I. Mani, R. Macmillan, S. Luperfoy, E. Lusher, and S. Laskowski. 1996. Identifying unknown proper names in newswire text. In *Proceedings of the ACL SIG Workshop on Acquisition of Lexical Knowledge from Text*.

J. Mayfield, P. McNamee, and C. Piatko. 2003. Named entity recognition using hundreds of thousands of features. In *Proceedings of the Conference on Computational Natural Language Learning*.

V. Mittal, S. Baluja, and R. Sukthankar. 1999. Applying machine learning for high-performance named-entity extraction. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics*.

H. Nakagawa and T. Mori. 2002. A simple but powerful automatic term extraction method. In *Proceedings of the International Conference On Computational Linguistics (COLING)*.

P. Osenova and S. Kolkovska. 2002. Combining the named-entity recognition task and np chunking strategy for robust pre-p. In *Proceedings of the Workshop on Linguistic Theories and Treebanks*. Sozopol, Bulgaria.

W. Wong. 2005. *Practical Approach to Knowledge-based Question Answering with Natural Language Understanding and Advanced Reasoning*. Master's thesis, National Technical University College of Malaysia.

A case study of comparison of several methods for corpus-based speech intention identification

Kazutaka Shimada **Kaoru Iwashita** **Tsutomu Endo**
Department of Artificial Intelligence,
Kyushu Institute of Technology
680-4 Iizuka Fukuoka 820-8502 Japan
{shimada, k_iwa, endo}@pluto.ai.kyutech.ac.jp

Abstract

Dialogue analysis is one of the most important tasks for human-machine interaction. It is important for dialogue systems to recognize the intention of an utterance and the intentional structure of a discourse. If a system identifies the intentions of each utterance in a dialogue correctly, it can recognize the structure of the dialogue. This paper describes methods for intention identification of an utterance. In this paper we apply two methods to the task and evaluate the performance. The 1st method is based on a similarity measure between an input utterance and utterances in the corpus (Case examples). We compare several similarity measures in the experiment. The 2nd method is based on the Maximum Entropy (ME) method. We compare our methods with related work. In the experiment, the ME method produced the best performance. For the coverage, the similarity based method with the inner product outperformed the ME method. In addition, we verify the effect of dependency relations between words and context information for this intention identification task.

1 Introduction

Dialogue analysis is one of the most important tasks for human-machine interaction. Speech understanding systems have been developed to practical use recently. However, speech understanding is just one component of dialogue understanding systems. It is important for dialogue systems to recognize the intention of an utterance and the intentional structure of a discourse. Recognition of them leads to generation of more appropriate model of plans for discourse and a solution of a problem (Higashinaka et al., 2003, 2005). If a system estimates the intentions of each utterance in a dialogue correctly, it can recognize the structure of the dialogue. Some researchers have reported rule-based or example-based intention understanding methods (Kimura et al., 1998; Matsubara et al., 2002; Irie et al., 2003, 2004; Inui et al., 2003). Kimura et al. (1998) have reported a rule-based method for intention understanding. In general, constructing rules for the method, however, is costly. Irie et al. (2003) have proposed an example-based method for the task. They used a similarity measure and a sequence of utterances for the intention understanding. Inui et al. (2003) have proposed a method for classification of open-ended questionnaire texts based on surface expressions. They used the Maximum Entropy method for the task.

In this paper we describe two methods for intention identification of an utterance. First, we explain a method based on a similarity measure. In this method we compute a similarity between an input utterance and utterances in a corpus (case examples). We compare several similarity measures. Next, we describe a method based on the Maximum Entropy (ME) method. In addition, we discuss features for the methods and the effect of context information.

In Section 2, we explain our task, corpus and intentions. We prepare a manual for the tagging process of the intentions and evaluate the reliability of the tagging based on the manual. In Section 3, we describe the two methods for the intention identification. In Section 4, we evaluate the performance of each method and conclude this paper in Section 5.

2 Intention and Corpus

In this section we explain the construction of a corpus for our method and intentions in our task. Furthermore, we evaluate the validity of a tagging process for the corpus.

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 255–262,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

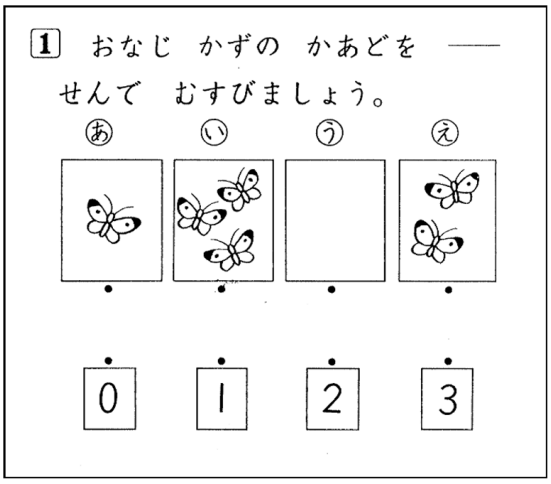
Table 1: The tagged corpus.

	# of turns	# of utterances
Teacher (T)	1390	1509
Student (S)	1449	1651
Total	2839	3160

2.1 Corpus

In this paper, we focus on a problem solving and knowledge acquisition system based on co-reference between drill texts and dialogue with a teacher (Endo and Kagawa, 1999; Shimada et al., 2007). We handled this task as a domain.

We collected a corpus consisting of simulated dialogues for this domain. The corpus consist of 81 dialogues with 27 test subjects for person-to-machine. The simulated dialogues for person-to-machine were collected by Wizard of OZ method (Fraser and Gilbert, 1991). We transcribed the dialogues manually, and then tagged fillers and ill-formed expressions, such as self-correction and hesitation, in them. Figure 1 shows an example of a drill text and a dialogue. The corpus contains 3160 utterances in 2839 turns (See Table 1). The turn denotes "S_i" and "T_i" in Figure 1.




```
S1: nanto onazi kazudesuka.
    (What is the number equal to?)
T1: kono waku no naka no chou wo kazoetekudasai.
    (Count the number of butterflies in this frame)
S2: kore wo kazoerunodesune.
    (May I count them?)
T2: soudesu.
    (yes)
S3: dokokara kaado wo erabunodesuka.
    (Where may I select from?)
T3: kokokara desu.
    (From here.)
S4: kono chou to kono kaado wo musubunodesune.
    (May I link card(1) and butterfly(1)?)
    .....
    ..... to be continued
```

Figure 1: An example of a drill text and a dialogue.

2.2 Intention

Various intention tags have been proposed in related studies (Araki et al., 1999; Irie et al., 2006; Searle, 1969; Walker and Passonneau, 2001). Walker and Passonneau (2001) have proposed a dialogue act tagging scheme for a travel domain. Their tags consisted of three dimension: (1) a speech act dimension, (2) a task-subtask dimension, and (3) a conversational domain dimension. Irie et al. (2006) have described a design of speech intention tags for an in-car spoken dialogue corpus. The tag set contained four layers: (1) Discourse act, (2) Action, (3) Object, and (4) Argument layers. These intention tags are specialized for a task. As a result, their tags support to determine an operation of a speech understanding system. However, most of the intention tags designed by them are domain-specific tags.

On the other hand, there are familiar speech-act labels and intentions, such as "request", "assert" and "question" (Araki et al., 1999; Searle, 1969). These intentions are more abstract than the domain-specific intentions. The purpose of our study in this paper is comparison of several methods for identifying the intention of an utterance. Therefore we use the traditional intention tags and expand them in this paper.

For the corpus we tag an intention label for each utterance. The number of intention labels is 12. Some intentions possess an attribute. The 5 intentions out of 12 possess attributes. The total number of intentions including attribute patterns is 18. Table 2 shows the intention tag set in our task.

We prepare a manual for the tagging process. The manual contains the explanation of the intention labels, examples of utterance-intention pairs and the guideline of the tagging. In this paper we tag one intention label to one utterance. Figure 2 shows an example of utterances and their intention tags.

2.3 Evaluation of the Corpus

We need to consider the reliability of the tagging process. For the evaluation the Kappa coefficient (κ) has been adopted as a standard in the dialogue processing (Cohen, 1960). The Kappa coefficient is computed as follows:

$$K = \frac{P(O) - P(E)}{1 - P(E)} \quad (1)$$

Table 2: Intention tags.

Label	Attribute	Definition
INFREQ (INFormation REQuest)	Howto	Request for how to solve a problem
	Request	Request for information to solve a problem
INFSUP (INFormation SUPplement)	Howto	Supplement for how to solve a problem
	Confirm	Supplement for information to solve a problem
AGREXP (AGReement EXPression)	Correct	Agreement for a confirmation request
	Response	Agreeable response
ACTREQ (ACTion REQuest)	Solve	Request for an action to solve a problem
	Inform	ACTREQ:solve with some hints
	Other	ACTREQ except above
CONREQ (CONfirmation REQuest)	Inform	Request for confirmation of information to solve a problem
	Correct	Request for confirmation: right or wrong
ACTREP (ACTion REPort)	——	Report for an action request
DISEXP (DISagreement EXPression)	——	Expression of disagreement
BEGDIA (BEGinning of DIAlogue)	——	Beginning of a dialogue
ENDDIA (END of DIAlogue)	——	End of a dialogue
REQRES (REQuest for REStating)	——	Request for restating an utterance; e.g. paraphrasing
ACTRES (ACTion REStating)	——	Action restating an utterance
OTHER	——	An insignificant utterance; e.g. filler only

Table 3: Kappa coefficient.

	Experience	NoExperience
$P(O)$	0.851	0.745
$P(E)$	0.027	0.020
Kappa coefficient	0.846	0.740
Average	0.793	

where $P(O)$ is the proportion of times the annotators agree and $P(E)$ is the proportion of times that we would expect the annotators to agree by chance. If the Kappa coefficient is close to 1, the degree of agreement is high.

We evaluated our tagging process with 10 test subjects. 5 persons out of 10 have tagging experience for other utterances in our task. For this evaluation we extracted 6 dialogues (210 utterances) from the corpus randomly. Table 3 shows the results of the evaluation. We obtained $\kappa = 0.793$ on average. In general $\kappa > 0.7$ indicates the substantial agreement of the tagging process (Araki et al., 1999). The result shows the high reliability of the tagging process with our manual.

3 Identification of Intention

In this section we explain two methods for intention identification: similarity measures and the maximum entropy method.

3.1 Similarity Measures

We compute a similarity between an input utterance and utterances in a corpus. We have already compared nine similarity measures: the inner product, some versions of the cosine measure, the Dice coefficient and the Jaccard coefficient (Iwashita et al., 2007) . In this paper we use three similarity measures that obtained high accuracy in the nine similarity measures. The similarities are the inner product (Inn), the Dice coefficient (Dice) and correspondence of morphemes (CorM). The correspondence of morphemes (CorM) is a similarity measure reported in related work (Irie et al., 2003). These similarities are computed as follows:

$$Inn(U_x, U_y) = \sum_{i=1}^{T} x_i \cdot y_i \qquad (2)$$

S_i: Dou sureba ii desuka? :: INFREQ:Howto
(How should I do?)

T_i: Mazu chou no kazu wo kazoete kudasai. :: ACTREQ:Solve
(First, count the butterflies.)
Soshite onazi kazu no kaado wo sagashite kudasai :: ACTREQ:Solve
(Then, search the card containing the same number.)

S_{i+1}: ..... to be continued.

Figure 2: An example of utterances and their intentions.

where U is an utterance. x_i and y_i are the value of a word i in U respectively. For the inner product, the value is a binary indicator, namely 1 (i exists in U) or 0 (otherwise). T is the number of vectors.

$$Dice(U_x, U_y) = \frac{2\sum_{i=1}^{T} x_i \cdot y_i}{\sum_{i=1}^{T} x_i^2 + \sum_{i=1}^{T} y_i^2} \quad (3)$$

For the Dice coefficient, we use word frequency for x_i and y_i in U_x and U_y.

$$CorM(U_x, U_y) = \frac{2M_{xy}}{M_x + M_y} \quad (4)$$

where M_x and M_y are the number of morphemes in U_x and U_y. M_{xy} denotes the number of morphemes matched between U_x and U_y. This computation is the binary vector version of the Dice coefficient.

The identification process with the similarity measures is as follows:

1. divide utterances into words by using the morphological analyzer ChaSen (Matsumoto et al., 1999)
2. construct a vector space for the identification process
3. compute the similarity between an input utterance (U_x) and each utterance in a corpus (U_y)
4. decide by a majority vote if some intentions possess the same maximum similarity.

3.2 Maximum Entropy

Maximum entropy modeling (ME) is one of the best techniques for natural language processing (Berger et al., 1996). The principle of the ME is expressed as follows:

$$P_\Lambda(c|d) = \frac{1}{Z_\Lambda(d)} \exp\left(\sum_i \lambda_{i,c} f_{i,c}(d,c)\right) \quad (5)$$

$$Z_\Lambda(d) = \sum_{d,c} \exp\left(\sum_i \lambda_{i,c} f_{i,c}(d,c)\right) \quad (6)$$

where $Z_\Lambda(d)$ is a normalization function. $\Lambda = \{\lambda_1, ..., \lambda_n\}$ are parameters for the model. These parameters denote weights and significance of each feature. The parameter values are a set that maximizes the entropy concerning the classifier. $f_{i,c}(d,c)$ is a feature function that is defined as follows:

$$f_{i,c}(d,c') = \begin{cases} 1 & \text{if } exist(d,i) > 0 \text{ and } c' = c \\ 0 & otherwise \end{cases} \quad (7)$$

where $exist(d,i)$ is a indicator function. The value is 1 in the case that a feature i exists in a document d.

In this paper we use Amis, which is a parameter estimator for maximum entropy models[1]. We estimate parameters by using the generalized iterative scaling algorithm.

The identification process with the ME is as follows:

1. divide utterances into words by using the morphological analyzer ChaSen
2. extract features from a corpus and then construct a training dataset
3. estimate parameters by using the ME method
4. output the intention that possesses the maximum probability computed from the estimated model

3.3 Features

We employ two types of features for the vector space model: (1) Bag-of-words (BOW) and (2)

[1] http://www-tsujii.is.s.u-tokyo.ac.jp/amis/index.html

Table 4: Accuracy.

	Similarity			ME
	Inn	Dice	CorM	
Q1	80.23	81.53	83.21	**84.98**
Q2	90.43	88.72	89.11	**92.32**
Q3	89.13	88.43	89.14	**92.42**
Ave	86.60	86.22	87.16	**89.91**

Table 5: Coverage.

	Similarity			ME
	Inn	Dice	CorM	
Q1	**88.74**	84.38	86.19	85.21
Q2	**96.01**	92.59	93.00	92.32
Q3	**93.96**	90.11	90.82	92.42
Ave	**92.92**	89.03	90.00	89.98

Dependency trees. We use all words as vectors for the BOW. For the dependency trees, we analyze utterances by using Japanese dependency analyzer Cabocha (Kudo and Matsumoto, 2002). We use frequent sub-trees for the vector space. We employ the FREQT algorithm (Asai et al., 2002) to extract them. The FREQT is an efficient pattern mining algorithm for discovering all frequent tree patterns from a large collection of labeled ordered trees. It is based on the rightmost expansion, a technique to grow a tree by attaching new nodes only on the rightmost branch of the tree.

4 Experiments

In this section, we evaluate the proposed method with our corpus. In this experiment, we evaluated the following points: (1) comparison of the similarity-based and the ME methods, (2) the coverage rates of each method, (3) the effectiveness of the sequence of utterances and (4) the effectiveness of dependency trees.

We evaluated this task with 81-fold cross validation, namely leave-one-out cross-validation. In other words, we used 1 dialogue as test data and 80 dialogues as case examples from 81 dialogues in our corpus. In this experiment we used the FREQT system implemented by T. Kudo[2].

First we compared the similarity measures and the ME-based methods. Table 4 shows the experimental result. The ME-based method outperformed the similarity-based methods. For the similarity-based methods, there was no significant difference in them although the accuracy of the related work (CorM) was a little better than those of the inner and the dice similarities.

However, the similarity-based methods often contained several intentions that possessed the same maximum similarity. We focused on the coverage rate of each method. The coverage was computed as follows:

$$Coverage = \frac{IncCorrect}{\mathrm{N}} \tag{8}$$

where N is the number of utterances in test data. $IncCorrect$ is the number of utterances that contained the correct intention at least one. Table 5 shows the experimental result. Since the ME method usually output an unique intention as the 1st result, the difference between the accuracy and the coverage was slight. For this criterion, the inner product produced the best performance. This result denotes that the method with the inner product contains a possibility that it outperforms the ME method essentially.

Regarding all the methods, the accuracy of the problem Q1 was lower than those of Q2 and Q3. On our observation, this tendency depends on the difficulty of a problem. Q2 and Q3 were a simple problem, such as counting only. However Q1 needed to combine some processes[3]. To solve a difficult problem, a teacher and a student usually require enough interaction. Also, utterances in the dialogue often tend to be long and complex sentences. As a result, the accuracy of intention identification decreased.

In this paper we employed a majority vote for the final output of each similarity-based method. However, the frequencies of each intention tag in a corpus are not equable. For example, the tag <AGREXP:Response> in the corpus was 1/10 of the <AGREXP:Correct> tag. This denotes that the method probably outputs the <AGREXP:Correct> even if both of them exist in the intentions estimated by it. Therefore we introduced a normalization factor for the voting process.

$$v(int) = \frac{\#\text{ of } int \text{ in the output}}{\#\text{ of } int \text{ in the corpus}} \tag{9}$$

[2] http://cl.aist-nara.ac.jp/~taku-ku/software/freqt/

[3] Actually Figure 1 was the Q1 in this experiment. To solve this problem, a student needs at least 3 steps: counting butterflies, detecting the same number, and linking the cards.

Table 6: The use of a normalization factor for Inn.

	Majority	Eq. (9)
Q1	80.23	45.30
Q2	90.43	62.36
Q3	89.13	55.61
Ave	86.60	54.42

Table 7: The utilization of tag sequence.

	Inn		ME	
Use	OFF	ON	OFF	ON
Q1	**80.23**	80.12	**84.98**	82.51
Q2	**90.43**	88.16	**92.32**	90.33
Q3	**89.13**	89.04	92.42	**92.55**
Ave	**86.60**	85.77	**89.91**	88.47

where *int* is an intention. Table 6 shows the experimental result. The normalization factor was not effective although it was effective for some intentions, such as <AGREXP:Response>.

One of the solutions for this problem, i.e. low frequency intention tags, is utilization of context information, such as tag sequence. Next, we evaluated the effectiveness of the sequence of utterances. Generally, the intention of an utterance depends on the intentions of the previous utterances. The history of intentions is one of the most important context in a dialogue. Matsubara et al. (2002) and Irie et al. (2003) have reported the effectiveness of context information, i.e. intention n-gram probability.

We applied the intention tag of the adjacent utterance in a corpus to our methods. First we extracted intention patterns that are the same as the previous intention of the input utterance. Next we computed the similarity between the input and the utterances extracted from a corpus. We computed a similarity measure (or a probability for ME) for all utterances in the corpus if no intention patterns exist in the corpus. Finally we decided by a majority vote if some intentions that possess the maximum similarity exist.

For the ME method, we generated a pair which consists of the intention of the current utterance and the intention of previous utterance. We regarded the pair as a new intention. For example, assume that the intentions of U_{i-1} and U_i are REQRES and ACTRES respectively. In this situation, we consider the intention of U_i to be REQRES-ACTRES. We computed the maximum entropy model by using these pair tags. In this experiment we substituted this method for the extraction of candidates, namely utilization of a tag sequence.

Table 7 shows the experimental result. In the table, "ON" denotes that we used the tag sequence for the method. In this experiment, we employed the inner product as the similarity measure. As a result, the utilization of tag sequence was ineffective for our dataset[4]. The accuracy decreased because utilizing a tag sequence led to the decrease of candidates for the similarity calculation[5]. One of the reasons is that our intention tags were simpler than that of the related work (Irie et al., 2003). Their intention tags consist of four hierarchized relations. Although the accuracy for intention identification including lower-level intention, namely the argument layer, increased by using context information, that for only the highest level intention did not improve. Our intention tags are similar to the highest level intention, namely the discourse act, in the related work. Therefore utilization of context information for this experiment did not lead to the improvement of the accuracy.

Finally we evaluated the effectiveness of dependency trees. We use the inner product with binary vectors as the similarity measure. We compared two patterns: the minimum support and length of trees. The patterns are as follows:

Deps 1 the minimum support: 5, the pattern length: 2 to 5.

Deps 2 the minimum support: 20, the pattern length: 2 to 5.

The results are shown in Table 8. The result shows that using dependencies was ineffective. For several dialogues, the method with dependencies, however, outperformed that with the BOW only. Therefore we need the detail error analysis to obtain higher accuracy.

5 Discussion and Conclusions

In this paper we described a method for identification of the intention of an utterance. Our methods

[4]Needless to say, it was effective for some intentions, such as <AGREXP:Response>. However they did not lead to the improvement of accuracy because they were minorities in the test data

[5]In this experiment, the coverage rate also decreased. The coverage of the method with the inner product and context information was 88.67%. The original coverage was 92.92%. See Table 5.

Table 8: Accuracy with dependencies.

	BOW	Deps1	Deps2
Q1	80.23	80.31	**80.99**
Q2	**90.43**	89.29	89.53
Q3	**89.13**	88.45	88.12
Ave	**86.60**	86.21	86.02

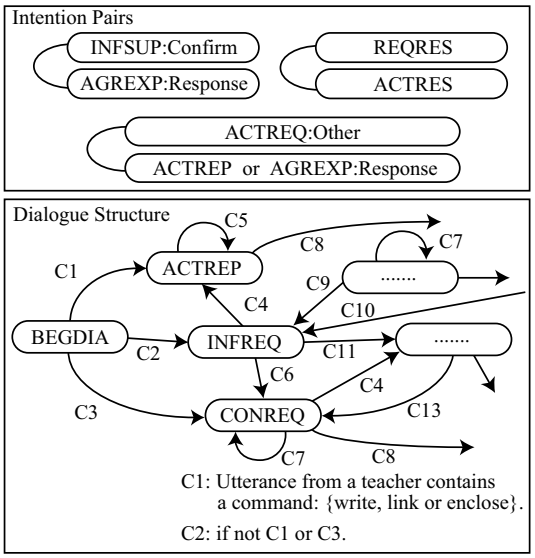


Figure 3: Intention pairs and A dialogue structure in our corpus (Part of).

computed a similarity (or a probability) between an input utterance and utterances in a corpus. In this experiment, the Maximum Entropy method outperformed similarity-based methods. However, the similarity-based method with the inner product sometimes contained the correct intention in the output. This denotes a possibility that the method with some adjustments might improve the accuracy. For the final output of similarity-based methods, we employed a majority vote. Although we used a normalization factor for the voting process, it led to the decrease of the accuracy. In the experiment, tag sequence and dependencies were not effective for the similarity calculation. However, they are effective for some intentions. Therefore we need to consider the usage of them.

One of the solutions to improve the performance is utilization of dialogue/discourse structures. Kato et al. (2005) have reported a dialogue annotation of an in-car speech corpus. They expressed the dialogue structure as a binary tree. In our task, intentions also contain a relation between them: a pair relation. We are developing a dialogue structure based on the analysis of our corpus. The dialogue structure is expressed by a transition network. Figure 3 shows a part of the relations and the dialogue structure The network contains the condition for the transition. We think that the relations and the structure are more useful than the context information that was described in Section 5 because they are global constraints in the dialogue. Effective utilization of relations between intention labels is one future work.

Table 9: Accuracy with a small corpus.

	Inn	Dice	CorM
Q1	76.47	74.79	76.38
Q2	85.10	85.87	86.02
Q3	84.67	84.78	84.13
Ave	82.08 (-4.52)	81.81 (-4.41)	82.18 (-4.98)

One of the approaches for the improvement of the accuracy is to incorporate high-level/abstract knowledge and heuristics. Matsubara et al. (2002) have used a word class for the similarity calculation. Also, in Japanese, clues of some intentions often appear in the end of the sentence. To weight words of the sentence-end might serve an important role in this task. Furthermore, we need to compare the method in this paper with other similarity measures and statistical techniques, such as decision tree learning (Irie et al., 2004) and Support Vector Machines (Vapnik, 1999). Integration of several similarity measures and machine learning methods is one of exciting approaches, such as the boosting algorithm (Freund and Schapier, 1996).

For our method, the size of the corpus is one of the most important problem. We evaluated the methods in the paper with a small dataset consisting of 27 dialogues. The result is shown in Table 9. The values in the parentheses in the table denote the difference of the accuracy from 81 dialogues (See Table 4). The accuracy decreased in the case that the corpus was small. Hence, in future studies, we need to develop a corpus tool kit for constructing a large corpus. Evaluation for other domains with the proposed method is also our future work.

References

M. Araki, T. Kumagai T. Ito, and M. Ishizaki. 1999. Proposal of a standard utterance-unit tag-

ging scheme (in Japanese). *Journal of JSAI*, 14(2):251–260.

T. Asai, S. Kawasoe K. Abe, H. Arimura, H. Sakamoto, and S. Arikawa. 2002. Efficient substructure discovery from large semi-structured data. In *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM'02)*, pages 158–174.

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):39–71.

J. Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

T. Endo and T. Kagawa. 1999. Cooperative understanding of utterances and gestures in a dialogue-based problem solving system. *Computational Intelligence*, 5(2):152–169.

N. Fraser and G. Gilbert. 1991. Simulating speech systems. *Computer Speech and Language*, 5(1):81–99.

Y. Freund and R. E. Schapier. 1996. Experiments with a new boosting algorithm. In *Proceedings of ICML*, pages 148–156.

R. Higashinaka, M. Nakano, and K. Aikawa. 2003. Corpus-based discourse understanding in spoken dialogue systems. In *Proceedings of ACL2003*, pages 240–247.

R. Higashinaka, K. Sudoh, and M. Nakano. 2005. Incorporating discourse features into confidence scoring of intention recognition results in spoken dialogue systems. In *Proceedings of ICASSP2005*, volume 1, pages 22–28.

H. Inui, M. Murata, M. Uchiyama, and H. Isahara. 2003. Classification of open-ended questionnaire texts based on surface expressions (in Japanese). *Journal of Natural Language Processing*, 10(2):19–42.

Y. Irie, S. Matsubara, N. Kawaguchi, Y. Yamaguchi, and Y. Inagaki. 2003. Speech intention understanding based on spoken dialogue corpus (in Japanese). In *SIG-SLUD-A301-03*, pages 7–12.

Y. Irie, S. Matsubara, N. Kawaguchi, Y. Yamaguchi, and Y. Inagaki. 2004. Speech intention understanding based on decision tree learning. In *Proceedings of the 8th International Conference on Spoken Language Processing*.

Y. Irie, S. Matsubara, N. Kawauguchi, Y. Yamaguchi, and Y. Inagaki. 2006. Layered speech-act annotation for spoken dialogue corpus. In *Proceedings of 5th International Conference on Language Resources and Evaluation*, pages 1584–1598.

K. Iwashita, K. Shimada, and T. Endo. 2007. Case-based estimation of speech intention (in Japanese). In *HINOKUNI symposium 2007 CD-ROM A-3-2*.

S. Kato, S. Matsubara, Y. Yamaguchi, and N. Kawaguchi. 2005. Dialogue structure annotation of in-car speech corpus based on speech-act tag. In *Proceedings of International Conference on Speech Databases and Assessment*, pages 159–163.

H. Kimura, M. Tokuhisa, K. Mera, K. Kai, and N. Okada. 1998. Comprehension of intentions and planning for responses in dialogue (in Japanese). In *IEICE Technical Report of Thought and Language, TL98-5*, pages 25–32.

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning 2002*, pages 63–69.

S. Matsubara, S. Kimura, N. Kawaguchi, Y. Yamaguchi, and Y. Inagaki. 2002. Example-based speech intention understanding and its application to in-car spoken dialogue system. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 633–639.

Y. Matsumoto, A. Kitauchi, T. Yamashita, Y. Hirano, H. Matsuda, , and M. Asahara. 1999. Japanese morphological analysis system chasen version 2.0 manual 2nd edition. Technical report, NAIST.

J. Searle. 1969. *Speech act*. Cambridge University Press.

K. Shimada, T. Endo, and S. Minewaki. 2007. Speech understanding based on keyword extraction and relation between words. *Computational Intelligence*, 23(1):45–60.

V. N. Vapnik. 1999. *Statistical Learning Theory*. Wiley.

M. Walker and R. Passonneau. 2001. Date: A dialogue act tagging scheme for evaluation of spoken dialogue systems. In *Proceedings of Human Language Technology Conference*, pages 66–73.

Author Profiling for English Emails

Dominique Estival, Tanja Gaustad, Son Bao Pham, Will Radford
Appen Pty Ltd
Chatswood NSW 2067, Australia
{destival,tgaustad,sbpham,wradford}@appen.com.au

Ben Hutchinson*
Google Australia
Sydney NSW 2000, Australia
benhutch@google.com

Abstract

This paper reports on some aspects of a project aimed at automating the analysis of texts for the purpose of author profiling and identification. The complete analysis provides probabilities for the author's basic demographic traits (gender, age, geographic origin, level of education and native language) as well as for five psychometric traits. We describe the email data which was collected for the project, the ways this data is processed and analysed, and the experimental setup used for classification with the Text Attribution Tool (TAT) before presenting our results for the demographic and psychometric traits using English email. Results are very promising for all ten traits examined.

1 Introduction

Automatically predicting the identity of authors from their texts has a number of potential applications. For example, if a text poses any type of threat, then identifying the source of the threat is the first step in countering it. In this context, author profiling forensics can be helpful to at least narrow the list of potential authors (Corney et al., 2002; Argamon et al., 2005; Abbasi and Chen, 2005). Another area where author identification and profiling can provide valuable information is in deriving marketing intelligence from the acquired profiles (Glance et al., 2005) and the rapidly growing field of sentiment analysis and classification (Oberlander and Nowson, 2006).

In this paper, we present some aspects of a research project aimed at automating the analysis of text in various forms of data (email in the first instance), for the purposes of author profiling and identification. The complete analysis provides probabilities for the author's basic demographic traits such as gender, age, geographic origin, level of education and native language, as well as for five psychometric traits. The prototype software[1] also provides a probability of a match with other texts, both from known and unknown authors. This paper describes experiments on profiling authors of emails in English using various machine learning (ML) algorithms and feature sets.

We will first give an overview of previous work done in authorship and text attribution and our specific take on the problem (Section 2). In Section 3, we introduce the specifics of our data set, including information on the data collection process, normalisation and validation as well as the in-depth analysis performed on the data. After a description of the experimental setup in Section 4, we report on and discuss our results for five demographic and five psychometric traits in Section 5. Section 6 concludes the paper with some pointers to future experiments.

2 Author attribution and author profiling

Authorship attribution is the task of deciding for a given text which author (usually from a predefined set of authors) has written it. Classic examples include authorship attribution studies on the Bible (Friedmann, 1997), Shakespeare's works (Ledger and Merriam, 1994) or the Federalist Papers (Mosteller and Wallace, 1964). For a long time, the main applications have been restricted to literary texts. Recently, authorship attribution has gained new life in the fight against cyber crime and in a more general search for reliable identification techniques (Abbasi and Chen, 2005; de Vel et al.,

*The work presented in this paper was carried out while this author was working at Appen Pty Ltd.

[1] The working prototype of TAT has now been completed and delivered, but will not be discussed here in more detail.

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 263–272,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

2001, 2002; Zheng et al., 2003).

The task of authorship attribution has traditionally been carried out on data from small sets of authors. For larger data sets, involving more authors, the challenge of identifying individual authors is more difficult. In such cases, predicting characteristics, or traits, of authors can be a good alternative and provide clues as to the author's identity.

Author profiling is the task of predicting one or more such author traits and an author profile consists of the resulting set of one or more predicted traits. Importantly, and contrary to author attribution, the author profiling task is possible even when documents by the author are not in the training data. Also in contrast to author attribution, greater accuracy can be expected when the training data contains texts from more authors, because the models of each trait are then expected to be more robust.

Most research into author profiling focuses on the prediction of a small number of traits, e.g. gender (Corney et al., 2002), gender and age (Koppel et al., 2006), neuroticism and extraversion[2] (Argamon et al., 2005), neuroticism, agreeableness, extraversion and conscientiousness (Oberlander and Nowson, 2006), neuroticism, agreeableness, extraversion, conscientiousness, and openness (Mairesse and Walker, 2006). In contrast, the present study is the largest that we know of in terms of the number of traits predicted, ten in total (see Section 4).

Corney et al. (2002) describe an experiment in predicting gender in email with a machine learner, namely SVM. The majority of the features the authors use are similar to the ones employed in our system with the exception of the gender-preferential linguistic features. Overall, this approach satisfactorily discriminates between male and female authors. The main finding of Corney et al. is that function words provide the most important clues for differentiating gender. This study is most comparable to the work presented in this paper.

Argamon et al. (2005) try to distinguish high neuroticism from low neuroticism and extraversion from introversion in informal texts. The approach uses four sets of features (lexical features, conjunctive phrases, modality, appraisal) and an SMO machine learner for classification. Even though the authors define the task as a binary classification task distinguishing between the top third and the bottom third scores for the two psychometric traits, the results are inconclusive. The authors conclude that most probably the features chosen are not adequate for the task.

Koppel et al. (2006) predict gender and age in blog data. Due to the massive number of authors investigated (18,000), a regular classification approach is not feasible. The authors opted for an Information Retrieval technique using various term frequency-inverse document frequency weights in combination with a cosine measure for similarity. In approximately 70% of the attempted predictions, this method is not able to pick an author for a given blog. Prediction accuracy for the blogs that have been assigned an author reaches 88.2%. Because of the very different approach, the use of different features and the fact that in their study the two traits are evaluated in combination, it is impossible to compare this approach to ours.

Another blog study which predicts four psychometric traits (neuroticism, agreeableness, extraversion and conscientiousness) using machine learners (Naive Bayes and SVM) is presented in (Oberlander and Nowson, 2006). The blog corpus consists of 71 authors (notably smaller than the data used in (Koppel et al., 2006)) and the authors use word bi- and trigrams as features. They report on many different setups based on binary or n-ary classification and different levels of restrictions on feature selection. Of the tasks described, task 6 using a 3-way split and automatic feature selection is most similar to the setup used in our experiments. Oberlander and Nowson achieve promising results for all the traits examined.

Mairesse and Walker (2006) try to predict five psychometric traits in order to build reliable personal profiles for dialogue management. They use various machine learners on two corpora, one written and one spoken, with features based on speech acts (command, prompt, question, assertion), the Linguistic Inquiry and Word Count (LIWC) tool (Pennebaker et al., 2001) (e.g. ratio of pronouns, positive emotion words), the MRC Psycholinguistic database (Coltheart, 1981) (frequency of word use, familiarity, age of acquisition) and—for the spoken corpus—prosody (voice pitch and intensity, speech rate). The features used in their study are very different from those used for the exper-

[2]Both "extraversion" and "extroversion" can be used to describe this particular dimension of human personality. In the domain of personality models, however, "extraversion" is the accepted term.

iments described in this paper. Unfortunately, Mairesse and Walker (2006) do not include baselines in their paper and we can therefore not compare our performance to theirs. The main conclusions are that observed personality is easier to model than self-reports and that spoken language is easier to model than written language.

3 Data

The data we are using for the experiments described here are email messages which have been collected specifically for the purpose of the TAT project. Because this constitutes a new data set, we include a discussion of the data collection methodology, the normalisation and validation of the email messages before their inclusion in the corpus, as well as the analysis and more in-depth pre-processing of the actual text.

3.1 Data Collection and Validation

We collected emails in several varieties of English, including native and non-native speakers of English coming from different geographical areas. Table 1 gives an overview of the different types of emails written in English, with statistics for the number of authors, number of emails and total number of words in the corpus.

Respondents were contacted via a recruitment process which included notification of privacy and assurance that their identity would be protected. Respondents agreed to fill out a web questionnaire and to donate ten email messages. The questionnaire elicited information about both demographic and psychometric traits. Demographic traits cover basic demographic information about the author: age, gender, native language, level of education, and main country of residence. For the psychometric traits, all the English collections (with the exception of the Egyptian one) use a short version of the International Personality Item Pool questionnaire, consisting of 41 questions (Buchanan et al., 2005).

After completing the questionnaire, respondents forwarded previously sent emails, e.g. from their email client "SentBox", to the data collection email address. The raw email documents were stored on a dedicated mail server and from then on all further operations took place on copies. The email messages were first checked manually to filter out erroneous content such as foreign language emails or forwarded chain letters and to ensure consistency and accuracy of the documents in the corpus. Email messages containing less than 5 lines of text (at a fixed line length of 80 characters) were excluded from our data set.

Data from authors who did not meet a set of satisfaction criteria was also removed. These criteria were: 1) a valid questionnaire received for a given author; 2) at least 5 valid email messages for the author; and 3) a total word count for that author's valid email messages of at least 1000 words. An additional requirement was that the emails be from different domains, such as personal or business emails. In the end, 1033 respondents were validated with a combined total of 9836 email messages. This corresponds to 57% of the total number of emails collected and constitutes the data set used for the experiments reported in this paper.

3.2 Normalisation

Forwarded emails come in a variety of formats and pre-processing was required to normalise those. This includes resolving encoding issues, identifying the actual content part of the email message and removing artefacts introduced through the data collection methodology.

The process takes into account languages other than English and writing systems other than Latin script. Ultimately, all the email documents are stored in UTF-8 encoding. However, the original documents come in a variety of character set encodings, e.g. US-ASCII, UTF-8, ISO-8859-1. Moreover, some documents have inconsistent encoding declarations, e.g. a UTF-8 document may have a header claiming it is ISO-8859-1. Such cases are handled using a set of heuristics to guess at the correct encoding.

Parts of the MIME message containing the body of the email are distinguished from headers, attachments and forwarded material sent as embedded MIME messages. Non-text/non-HTML payload contents, or anything with a file name, are also stripped from the version of the documents to be processed.

3.3 Analysis

The aim of the analysis phase is to produce annotations which are later used during the feature extraction phase. The analysis stage consists of three modules: document parsing[3], text processing and

[3] In order to avoid confusion, it is important to stress that we are using "parsing" in the computer science interpretation of the term and not as in "syntactic parsing".

Collection	Native lang.	# authors	# emails	# words total	# words by author
United States	English	415	4,533	2,405,792	1,886,389
United Kingdom	English	23	273	178,400	137,238
Australia/New Zealand	English	133	1,387	513,065	437,454
United States	Spanish	174	1,823	519,504	461,767
Egypt	Arabic	288	1,820	451,903	444,325
Total		1,033	9,836	4,068,664	3,367,173

Table 1: Overview of the collected English email data.

linguistic analysis, each resulting in a different set of annotations.

There are two ways annotations can be created: automatically, by the analysis modules described below, and manually by human annotators according to pre-specified guidelines using the Callisto tool (Mitre, 2006). Our emphasis is on automatically created features, mainly using the manual annotation for validation purposes.

3.3.1 Document parsing

The purpose of the document parsing stage is to classify the body text in an email document into five categories:

1. Author text: text that was written by the author and that is not contained in an embedded reply chain of email messages;

2. Signature: email signature text, which typically includes contact information, professional details, and/or quotations;

3. Advertisement: advertisements automatically appended by the author's email client, such as Yahoo and Hotmail;

4. Quoted text: extended quotations, e.g. song lyrics, poems, newspaper articles;

5. Reply lines: text that was written in a previous email message that the author is either forwarding or replying to, including text by other writers, text in previous emails by the author of the current email, with their email signatures, advertisements and quotations.

The input is an email document and the output is the same email document, with the lines of the body text categorised into one of those five classes. Consecutive lines of the same class are covered by an annotation of the corresponding type. These annotations are used to calculate structural features.

The document parsing stage is crucial, as it is the linguistic features of the "author text" which provide the most clues for author attribution. Table 1 gives the number of words written by the author in the body of the email messages and the total number of words contained in the emails.

We experimented with an existing tool, Jangada (Carvalho and Cohen, 2004), which identifies signature blocks and replies, but unfortunately it did not perform very well on our data and we identified some shortcomings. The main issues were that we could not configure Jangada to distinguish the more detailed categories enumerated above nor include additional features in its (rather simple) statistical model of document structure. Also, in spite of what seems to be suggested in (Carvalho and Cohen, 2004), Jangada makes systematic errors and does not identify forwarded message text as reply lines.

The poor performance of Jangada led us to develop our own document parser. This document parser builds a statistical model of document structure by extracting features from each line of a document and using them to train a statistical classifier. Conditional Random Fields (Lafferty et al., 2001) have been shown to work well at labeling sequential data as they can effectively combine contextual information and line-specific features and have therefore been employed as a statistical model in our system.

To compare the performance of Jangada with that of our document parser, we used ten-fold cross validation on all of the English data. For each train-test partition, our document parser was trained on the training part and its performance was compared with Jangada's on the testing part. As Jangada can only identify *Author text*, *Reply* and *Signature* lines, we cannot compare the per-

formance on the task of recognising all five categories. For the task of identifying those three categories, our document parser achieved an accuracy of 88.16% while Jangada performed at 64.22%. When focusing on the task of identifying only author lines, our document parser reached an F-score of 90.76% compared to 74.64% from Jangada.

3.3.2 Text processing

Text processing consists of two stages: segmentation and punctuation analysis. First, the text in the email is split into paragraphs and paragraphs are split into sentences and tokens. The latter is currently performed with third party tools (Cunningham et al., 2002) which generate *paragraph*, *sentence* and *token* annotations respectively. The use of sentence punctuation marks and other special characters is then analysed, including but not limited to, special markers, e.g. two hyphens "- -" which often indicate that an email signature follows; quotation marks, which sometimes signal the presence of a quotation; and emoticons, such as ":-)" or ":o)". This information is stored as attributes of *token* annotations which are used for calculating character level features.

3.3.3 Linguistic analysis

The aim of the linguistic analysis stage is to produce more linguistically informed annotations, such as Part-Of-Speech (POS) tags. Linguistic processing deals with the linguistic, usually language-dependent, aspects of texts and it requires linguistic resources such as word lists.

To identify key phrases, we have developed a Named Entity recognizer using gazetteers and grammars, which identifies people, locations, organisations, dates etc. We implemented our own NE recognizer because most available systems were developed on news corpora, i.e. a very different domain from ours. All the heuristics in our recognizer are based on email data. Additional lexicons were developed manually to identify set phrases, for instance farewells and greetings.

4 Experimental setup

Many problems in NLP have lent themselves to solutions using statistical language processing techniques. Author profiling can be considered a type of document classification task, where the classes correspond to traits of the authors. These traits are arranged along various dimensions, with different options for each dimension being mutually exclusive. For example male and female are the possibilities for the gender dimension.[4] For each dimension, the email and questionnaire data are used to construct classifiers, using a range of ML techniques.

Each document constitutes a single data instance for the purposes of the experiments. For each experiment, ten-fold cross-validation was used, so the results reported in Section 5 are on the entire data set. We also used ten-fold cross-validation during training, for feature selection and model parameter tuning. This means that for each train-test partition in cross-validation, we tuned the parameters on the training part using ten-fold cross-validation. Once the best combination of ML classifiers, parameters and feature selection has been determined, that model is used to classify the test data to evaluate the performance of the chosen model.

4.1 Traits and classes

We distinguish five demographic and five psychometric traits in the experiments presented in this paper, namely *age*, *gender*, *native language*, *level of education* and main *country* of residence for the demographic traits, and *agreeableness*, *conscientiousness*, *extraversion*, *neuroticism* and *openness* for the psychometric traits (Matthews et al., 2003). This information is extracted from the questionnaire filled out by the respondents.

For the traits taking numerical values, subjects were split into three classes based on the first and third quartiles. Table 2 summarizes the data distribution for each trait across these classes.

4.2 Features

For each author, a feature vector is calculated. Typically, a feature is a descriptive statistic calculated from both the raw text and the annotations. For example, a feature might express the relative frequency of two different annotation types (e.g. number of words/number of sentences), or the presence or absence of an annotation type (e.g. signature).

For the English data, 689 features were calculated. These were divided into three main groups,

[4] It has been pointed out that a male author could also be writing from a female perspective and vice versa. In our corpus, however, these cases do not occur and therefore gender is treated as a binary variable.

Demographics				
Age:	Gender:	Native language	Level of education:	Country:
<25 (423)	Male (483)	English (571)	No tertiary edu. (498)	USA (528)
25 to 35 (350)	Female (550)	Arabic (288)	Some tert. edu. (535)	Egypt (288)
>35 (260)		Spanish (174)		AUS/NZ (133)
				Other (84)
Psychometrics				
Agreeableness:	Extraversion:	Neuroticism:	Conscientiousness:	Openness:
<5	<1	<-7	<3	<2
5 to 8	1 to 7	-7 to -2	3 to 9	2 to 7
>8	>7	>-2	>9	>7

Table 2: Traits and Classes, with frequencies in parentheses where applicable.

namely character-level, lexical, and structural features. In turn, we subdivided these main groups into feature groups which subsume all features of a particular type. An overview of the feature groups is shown in Table 3. The main purpose of these groupings was to make more informed choices during the feature selection stage and to facilitate experimentation with various combinations of feature groups. For instance "char+lexical" denotes the combination of all character-level and all lexical-level features, whereas "all-html" expresses that all features except the html group were used during classification.

Character-level features cover features such as the frequency of punctuation characters or word length. New types of features for this group include case-based features relating to occurrences of CamelCase and slow Shift release, e.g. "HEllo".

The lexical feature group comprises for instance function words and POS. A special addition to this group are features based on words that are highly correlated with one of the values for a particular author trait. For example, younger people use more first person pronouns whereas older people use more third person pronouns, and non-US people are more likely to use the abbreviations "u" and "ur" (for "you" and "your").

At the structural level, we included features such as paragraph breaks and the presence or absence of certain HTML tags. The annotations created by our document parser also produced features which identify interleaved author and reply text, as opposed to a consecutive text structure.

4.3 Classification algorithms and feature selection

The aim of the classifier is to match feature vectors from the document with author traits. Ordered pairs of feature vectors and author traits are used to train and tune machine learning classifiers. Formally, classifiers are functions which map feature vectors to author traits and there will be classifiers for each author trait such as gender, age, etc.

We apply various machine learning algorithms as classifiers, using the WEKA toolkit (Witten and Frank, 2005) to find the best classifier for each trait. During training, classifiers are created by the selection of sets of features for each author trait, and classifier parameters are tuned through cross-validation. To evaluate and test the classifiers, new documents are given as input and existing classifiers are selected to predict author traits.

The machine learning algorithms we tried include decision trees (J48 (Quinlan, 1993), RandomForest (Breiman, 2001)), lazy learners (IBk (Aha et al., 1991)), rule-based learners (JRip (Cohen, 1995)), Support Vector Machines (SMO (Keerthi et al., 2001), LibSVM (Chang and Lin, 2001)), as well as ensemble/meta-learners (Bagging (Breiman, 1996), AdaBoostM1 (Freund and Schapire, 1996)). These algorithms were used in combination with feature selection methods based on either a feature subset evaluator together with a search method (consistency subset evaluator with a best-first search) or a single attribute evaluator with various numbers of attributes selected (χ^2, GainRatio, and InformationGain) (see chapter 10.8 in (Witten and Frank, 2005) for details).

Main group	Feature group	Description
character		Features at character level
	case	Features using *case* attributes of characters
	wordLength	Features invoking word length
lexical		Features at lexical level
	functionWord	Features invoking function words
	correlate	Features using words that are highly correlated with a trait class
	namedEntities	Features using named entities
	POS	Features based on POS
structural		Features at structural level
	docCategory	Features specifying the category of an email (e.g. personal)
	html	Features pertaining to the html rendering of the email

Table 3: Feature groups.

5 Results and discussion

The results shown here were computed on the English email data set described in Section 3 using the different classifiers and general setup introduced in Section 4. Table 4 shows the results on all ten traits (demographic and psychometric). It also includes the respective baseline associated with each separate classification task, calculated on our data set of 9,836 emails. Furthermore, we state which settings (ML algorithm, feature selection, and feature set) were used to achieve the results reported. Education and gender are both binary classification tasks, whereas age and native language have three classes and country of residence four classes. All the psychometric traits are divided into three classes (see Section 4.1 and Table 2 for details on the exact split).

These results show that for the demographic and the psychometric author profile, classification is significantly[5] improved over the baseline for all ten traits. This demonstrates that the approach we took of combining ML algorithms, together with our particular feature set, is successful for binary as well as n-ary classifications on very diverse classification tasks.

No clear picture emerges as to which ML algorithms perform best over all the classification tasks. Our results seem to indicate that SMO works well for three out of five demographic traits, and IBk shows good performance for three out of five psychometric traits.

With regard to feature selection, our expectation was that for a Support Vector Machines algorithm, such as SMO, feature selection should not make a huge difference whereas an algorithm based on decision trees, such as RandomForest, would be more sensitive to feature selection in general. This expectation was, however, not confirmed during our experiments.

Looking at the features used for the predictions, taking all available features (with or without the exclusion of a certain feature group) works best for all demographic traits. On the other hand, all psychometric traits except openness rely on the combination of character-based and structural features; openness relies on structural features alone.

It is especially interesting to see that for the prediction of level of education, explicitly excluding function words from the features used leads to the best performance. Previous publications have found that function words are surprisingly effective on their own for authorship attribution (Argamon and Levitan, 2005; de Vel et al., 2002) or for certain traits, such as gender (Corney et al., 2002). However, in our data set, function words do not seem to work that well for predicting level of education

Comparing our findings to previously published results shows the following. For gender, our system performs very close to that of Corney et al. (2002): our relative error is 68.3 whereas their results yield 69.2 relative error. The results on psychometric traits reported in (Oberlander and Nowson, 2006) are very high and our system does not achieve the same performance, but one has to bear in mind that our corpus is 14 times larger than theirs and consists of email data rather than blogs. It is worth noting that on extraversion, our relative error slightly exceeds theirs.

Future research will need to investigate deeper

[5] All results are significant at p=0.01 using a χ^2 test.

Trait	ML algorithm	Feature Sel.	Best Features	Results	Baseline
Age:	SMO	–	all	56.46	39.43
Gender:	SMO	–	all	69.26	54.48
Language:	RandomForest	InfoGain	all-correlate	84.22	62.90
Education:	Bagging	–	all-functionWord	79.92	58.78
Country:	SMO	–	all	81.13	57.29
Agreeableness:	IBk	–	char+structural	53.16	40.51
Conscientiousness:	IBk	–	char+structural	54.35	43.72
Extraversion:	LibSVM	–	char+structural	56.73	45.17
Neuroticism:	IBk	–	char+structural	54.29	42.34
Openness:	RandomForest	–	structural	55.32	47.28

Table 4: Results for all demographic and psychometric traits on English email data.

features, such as syntactic information or writing style, which might help to classify the author traits more accurately. It would also be interesting to identify more specialised feature sets for each author trait. As mentioned above, a first step in this direction was made using words that correlate with a particular instantiation of a trait, but more research exploring features tailored to a given trait is necessary.

6 Conclusion and future work

We have presented some results of experiments to automatically predict author traits from email messages. This work is of interest for a number of potential applications, from threat identification to marketing intelligence. The results presented in this paper were conducted on the English subset (9836 emails) of the email data we have collected.

The experiments reported here were aimed at discovering how well a range of ML algorithms perform on our data set for five demographic and five psychometric author traits. Our results show that the chosen approach works well for author profiling and that using different classifiers in combination with a subset of available features can be beneficial for predicting single traits.

The next steps are to extend the text processing modules to other languages and to conduct a more thorough error analysis for our results. In particular, a more qualitative analysis per instance instead of overall performance will definitely provide more insight. In addition, it would be interesting to extend our approach to the task of author identification.

The email corpus presented in this paper has been collected in a principled manner and will hopefully become a valuable resource for the community.[6]

Acknowledgements

This research was carried out within the framework of a US Government BAA grant (IS-QD-2467). We would like to thank Judith Bishop for her efforts during data collection, as well as the anonymous ACL and PACLING reviewers for their valuable comments and insights on a previous version of this paper.

References

Ahmed Abbasi and Hsinchun Chen. 2005. Applying authorship analysis to Arabic web content. In Paul B. Kantor et al., editor, *Intelligence and Security Informatics, Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI 2005)*, pages 183–197. Springer.

David Aha, Dennis Kibler, and Mark Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Shlomo Argamon, Sushant Dhawle, Mosche Koppel, and James W. Pennebaker. 2005. Lexical predictors of personality type. In *Proceedings of the 2005 Joint Annual Meeting of the Interface and the Classification Society of North America*. St. Louis.

Shlomo Argamon and Shlomo Levitan. 2005. Measuring the usefulness of function words for authorship attribution. In *Proceedings of the 2005 ACH/ALLC Conference*. Victoria, BC.

[6] Please contact one of the authors at Appen for more information on the email corpus.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Tom Buchanan, John A. Johnson, and Lewis R. Goldberg. 2005. Implementing a five-factor personality inventory for use on the internet. *European Journal of Psychological Assessment*, 21:115–127.

Vitor Carvalho and William Cohen. 2004. Learning to extract signature and reply lines from email. In *Proceedings of the Conference on Email and Anti-Spam (CEAS-2004)*. Mountain View.

Chih-Chung Chang and Chih-Jen Lin. 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

William Cohen. 1995. Fast effective rule induction. In *Twelfth International Conference on Machine Learning*, pages 115–123.

Max Coltheart. 1981. The MRC psycholinguistic database. *Quarterly Journal of Experimental Psychology*, 33A:497–505.

Malcolm Corney, Olivier de Vel, Alison Anderson, and George Mohay. 2002. Gender-preferential text mining of e-mail discourse. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC 2002)*, pages 282–292. Las Vegas.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175.

Olivier de Vel, Alison Anderson, Malcolm Corney, and George Mohay. 2001. Mining email content for author identification forensics. *SIGMOD Record*, 30(4):55–64.

Olivier de Vel, Alison Anderson, Malcolm Corney, and George Mohay. 2002. E-mail authorship attribution for computer forensics. In Daniel Barbara and Sushil Jajodia, editors, *Data Mining for Security Applications*. Kluwer Academic Publishers.

Yoav Freund and Robert Schapire. 1996. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156. San Francisco.

Richard Friedmann. 1997. *Who wrote the Bible?* Harper, San Francisco.

Natalie Glance, Matthew Hurst, Kamal Nigam, Mathew Siegler, Robert Stockton, and Takashi Tomokiyo. 2005. Deriving marketing intelligence from online discussion. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 419–428. Chicago.

Sathiya Keerthi, Shirish Shevade, C. Bhattacharyya, and K. Murthy. 2001. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13(3):637–649.

Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Eran Messeri. 2006. Authorship attribution with thousands of candidate authors. In *Proceedings of SIGIR*, pages 659–660.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco.

Gerrard Ledger and Thomas Merriam. 1994. Shakespeare, Fletcher, and The Two Noble Kinsmen. *Literary and Linguistic Computing*, 9(3):235–248.

François Mairesse and Marilyn Walker. 2006. Words mark the nerds: Computational models of personality recognition through language. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society (CogSci 2006)*, pages 543–548. Vancouver.

Gerald Matthews, Ian Deary, and Martha Whiteman. 2003. *Personality Traits*. Cambridge University Press, Cambridge, second edition.

Mitre. 2006. Callisto. http://callisto.mitre.org/. Version 1.4.0.

F. Mosteller and D.L. Wallace. 1964. *Inference and Disputed Authorship: The Federalist*. Series in behavioral science: Quantitative methods edition. Addison-Wesley.

Jon Oberlander and Scott Nowson. 2006. Whose thumb is it anyway? Classifying author personality from weblog text. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 627–634. Sydney.

James Pennebaker, Martha Francis, and Roger Booth. 2001. LIWC: Linguistic inquiry and word count. `http://www.liwc.net/`.

Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, second edition.

Rong Zheng, Yi Qin, Zan Huang, and Hsinchun Chen. 2003. Authorship analysis in cybercrime investigation. In Hsinchun Chen et al., editor, *Proceedings of the first NSF/NIJ Intelligence and Security Informatics Symposium*, pages 59–73. Springer.

Towards a Semantic Analysis of Bahasa Indonesia for Question Answering

Septina Dian Larasati and Ruli Manurung
Faculty of Computer Science
University of Indonesia
Depok 16424, Indonesia
septina.dian@ui.edu, maruli@cs.ui.ac.id

Abstract

This paper presents a model of deep syntactic and semantic processing to support question-answering for Bahasa Indonesia. Starting from an existing unification-based grammar, we specify lexical semantics for each lexeme and semantic attachment rules for each grammar rule using lambda-calculus notation. This approach enables us to obtain semantic representations of Indonesian sentences in the form of first order logic literals. These representations are used by a question-answering module which stores declarative sentences as facts in a knowledge-base, and answers queries by unifying the question representation with known facts. We discuss an implemented prototype of this model using Prolog, and conclude with the remaining work that has to be done to realise our goal of a fully-fledged question-answering system for Bahasa Indonesia.

1 Background

In this section we present some information on the relevant issues to our research: question answering, specifically for Bahasa Indonesia, and existing semantic approaches. In Section 2 we give an overview of our system, before presenting all the details of the semantic analysis and representation of Indonesian sentences in Section 3. A Prolog implementation of this approach is discussed in Section 4, before we conclude by suggesting further avenues of work.

1.1 Semantic approaches to question answering

Question Answering, or QA, is a subtask of the more general Information Retrieval (IR) task. A QA system seeks to provide answers to questions expressed in natural language, where the answers are to be found in a given collection of documents. QA systems typically require more sophisticated linguistic analysis than conventional IR, as they need to reason about the types of questions, predicate argument structure, result aggregation, etc.

Most QA systems employ a pipeline composed of the following stages: first, the question is syntactically analysed to determine its type, and the type of the expected answer. Second, "shallow" statistical methods derived from the IR field, e.g. keyword-based retrieval, are used to retrieve a subset of documents that may potentially contain the answer. Finally, a finer-grained module processes this subset by focusing on specific passages and extracting candidate answers. QA systems often exploit data redundancy, where correct answers are more likely to appear multiple times in the corpus than false positives.

More recently, work has been done in developing QA systems that obtain semantic representations of both the query and (a subset of) the documents, e.g. Narayanan and Harabagiu (2004). The resulting logical forms abstract away purely syntactic variations from facts and queries, enabling more precise identification of answers to questions. They also capture semantic functions, which simple statistical systems that treat texts as bags-of-words fail to account for, e.g. the difference between *"USA invades Iraq"* and *"Iraq invades the USA"*. Additionally, semantic representations enable logical inference, allowing the QA systems to answer more complex queries by exploiting knowledge encoded in ontologies such as WordNet. Such an approach is exemplified by recent work in the textual entailment task, e.g. Bos and Markert (2006).

1.2 Question answering in bahasa Indonesia

Bahasa Indonesia (hereinafter simply 'Indonesian') is the official language of Indonesia, spoken

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 273–280,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

by over 100 million people. Given this fact, we believe it is underrepresented in terms of research into Indonesian QA, and Indonesian NLP in general.

There has been some work on developing QA systems for Indonesian, e.g. Wijono et al. (2006); Purwarianti et al. (2007). To our knowledge, all previous systems adopt a purely statistical method.

Wijono et al. (2006) sought to achieve multilingual QA by answering queries in Indonesian based on English documents. Questions are classified based on a manually constructed taxonomy of Indonesian questions. The query is then automatically translated into English using a commercial translator available online[1], and from then on is handled as a purely English QA task.

Purwarianti et al. (2007) uses a machine learning method to develop the question and answer classifier modules based on a corpus of raw text.

In this paper, we propose a model of an Indonesian QA system that adopts a deeper linguistic approach, leveraging a previously built syntactic parser for the Indonesian language (Joice, 2002).

2 Framework

The overall framework of our system consists of the following modules: a syntactic parser, a semantic analyser, and a question-answering module. We describe these three modules in the following subsections.

2.1 Syntactic parser

Our starting point was a computational account of Indonesian declarative sentences described in Joice (2002), which in turn is based on the official prescriptive grammar of formal Indonesian presented in Alwi et al. (1998). This grammar was developed using PC-PATR (McConnel, 1995), an implementation of the PATR-II formalism (Shieber, 1986).

Unfortunately, although it superficially used a constraint-based formalism, the grammar failed to elegantly capture regularities in Indonesian. For instance, various selectional restrictions were stipulated by explicitly enumerating all possible combinations of subcategories between, say, transitive verbs and their objects. Additionally, certain constructions best handled by recursive rules, e.g. adjectival phrases, were simulated up to a finite depth by explicit enumeration. This resulted in an explosion of rules once the grammar was transformed into a definite clause grammar, or DCG[2], to be used with our system (see Sect. 4).

Another drawback was that the system had a trivial account of morphology: the parser required a lexicon containing fully inflected words. Given the rich derivational morphology in Indonesian, this resulted in a very unwieldy dictionary.

Nevertheless, the grammar exhibited fairly wide coverage – see Joice (2002) for results of an experiment conducted with a corpus of sentences obtained from a national Indonesian newspaper.

To support our prototype question-answering application, we augmented the existing grammar with rules to account for interrogative sentences. More specifically, we handle both *in-situ* and *ex-situ* variations of wh-questions using *apa* (what), *siapa* (who), *mana* (where), *mengapa* (why), *kapan* (when), and *berapa* (how many/much).

2.2 Semantic analyser

The core of our research is concerned with the development of this module, i.e. a module that transduces semantic representations from parse trees. The semantic representations are designed to abstract away syntactic variations, allowing sophisticated automated processing of Indonesian texts.

We adopt a 'flat' semantic representation (Hobbs, 1985), details of which are presented in Sect. 3.1. Adopting the well-known rule-to-rule hypothesis (Bach, 1976), we decided to augment the lexicon with semantic information (Sect. 3.2), and develop semantic attachment rules for each grammar rule (Sect. 3.3).

Such a deeply handcrafted approach is at odds with the predominant approach in Information Retrieval of corpus-based statistical methods, which for much of the last decade has proved to be more successful at scaling up to the large resources available. However, we are not aware of similar previous work being done for Indonesian, and thus believe that our work establishes a foundation that will be of value for future research into, among others, Indonesian machine translation, dialogue systems, and natural language generation.

2.3 Question-answering module

Currently, we have implemented a prototype query processor in Prolog. The semantic representations

[1] `http://www.toggletext.com`

[2] a notation for CFGs with variable unification, commonly implemented in most Prolog systems

```
<expression>   :=   <literal>
<expression>   :=   <literal> ∧ <expression>
   <literal>   :=   <predicate>(<term>,<term>)
      <term>   :=   <variable>|<concept>
  <variable>   :=   [A-Z][a-z|A-Z|0-9]+
   <concept>   :=   [a-z][a-z|A-Z|0-9]+
 <predicate>   :=   [a-z][a-z|A-Z|0-9]+
```

Figure 1: BNF syntax of logical expressions

of Indonesian declarative sentences, i.e. as found within a collection of documents, are stored in a clausal knowledge base. Subsequently, the semantic representation of queries are transformed into Prolog rules which, when unified with the clause database, yields the appropriate answer. See Sect. 4 for details on this implementation.

3 Semantic representation

In this section we present all the details concerning the semantic representations of Indonesian sentences, i.e. the syntax of logical expressions, the content of lexical semantics, and how the semantic attachment rules are defined and applied. Our approach is reminiscent of Alshawi and van Eijck (1989).

3.1 Logical expressions

As mentioned in Sect. 2.2, we adopt a simple 'flat' semantic representation (Hobbs, 1985), where a logical expression is a conjunction of first order logic literals. The arguments of these literals represent domain concepts such as objects and events, while the functors state relations between these concepts. All variables are existentially quantified with the widest possible scope. The BNF syntax corresponding to our representation is given in Figure 1.

Additionally, following the approach in Durme et al. (2003), literals are divided into two categories, extrinsic and intrinsic literals. An **extrinsic literal** defines a relationship between two variables, whereas an **intrinsic literal** defines a relationship between a variable and its referent as being some semantic concept in some underlying ontology.

Table 1 shows a partial list of the intrinsic literals, whereas Table 2 shows a partial list of the extrinsic literals. Both types of literals are stored within the lexical semantics entries of the words that convey their meaning, which specify the Y variable (see Section 3.2).

Our semantic representation falls into the category of so called neo-Davidsonian approaches, where intrinsic literals are predicates over objects and events, and arguments and modifiers are specified via the thematic relations specified by the extrinsic literals.

$\lambda X time(X,Y)$	X is temporal object Y
$\lambda X location(X,Y)$	X is location object Y
$\lambda X object(X,Y)$	X is inanimate object Y
$\lambda X person(X,Y)$	X is animate object Y
$\lambda X event(X,Y)$	X is event "object" Y
$\lambda X property(X,Y)$	X is property "object" Y
$\lambda X quantity(X,Y)$	X is cardinal number Y
$\lambda X grade(X,Y)$	X is ordinal number Y
$\lambda X unit(X,Y)$	X is quantity unit Y

Table 1: List of some intrinsic literals

$\lambda X \lambda Y\ agent(X,Y)$	X is the agent of Y
$\lambda X \lambda Y\ patient(X,Y)$	X is the patient of Y
$\lambda X \lambda Y\ theme(X,Y)$	X is the theme of Y
$\lambda X \lambda Y\ attrib(X,Y)$	X is an attribute of Y
$\lambda X \lambda Y\ possess(X,Y)$	X possesses Y
$\lambda X \lambda Y\ purpose(X,Y)$	X is the purpose Y
$\lambda X \lambda Y\ cause(X,Y)$	X causes Y
$\lambda X \lambda Y\ manner(X,Y)$	X is a manner of Y
$\lambda X \lambda Y\ instrument(X,Y)$	X is an instrument of Y
$\lambda X \lambda Y\ condition(X,Y)$	X is a condition Y
$\lambda X \lambda Y\ result(X,Y)$	X is a result of Y
$\lambda X \lambda Y\ complement(X,Y)$	X is a complement of Y

Table 2: List of some extrinsic literals

3.2 Lexical semantics

Lexical entries of open class words are associated with exactly one intrinsic literal which asserts a reference to the domain concept the word is "about". The domain concepts can belong to any underlying ontology, e.g. WordNet (Fellbaum, 1998).

For example, given the WordNet verb synset labelled with the ID `s201132466`, whose gloss is "take in solid food", we can specify the following lexical semantics entries for *"memakan"* and *"menyantap"*, both Indonesian synonyms for the verb *"eat"*:

- *"memakan"*: $\lambda E \lambda A \lambda P\ \ event(E, \texttt{s201132466}) \wedge agent(E,A) \wedge patient(E,P)$
- *"menyantap"*: $\lambda E \lambda A \lambda P\ \ event(E, \texttt{s201132466}) \wedge agent(E,A) \wedge patient(E,P)$

Unfortunately, a mapping from WordNet synsets to an Indonesian lexicon does not currently exist – an ongoing project of ours aims at addressing this issue. Until then, we arbitrarily choose the root form of a synonym to act as the conceptual

symbol. For instance, the ID `s201132466` above is replaced with the token *makan*.

Note that the verbs *"memakan"* and *"menyantap"* above also specify the extrinsic literals $agent(E,A)$ and $patient(E,P)$. These are thematic relations that must be specified by complements within its syntactic projection. The variables will be subsequently bound with the variables of the subject and object through the lambda calculus operation of β-reduction (see Section 3.3 below).

In Alwi et al. (1998), there are several subcategories of nominals, e.g. temporal, location, object, person, etc. The lexical semantics of nominals is simply the appropriate intrinsic literal, e.g. the semantics of *"dapur"* (kitchen) is $\lambda X \;\; location(X, dapur)$ and the semantics of *"ayah"* (father) is $\lambda X \;\; person(X, ayah)$.

Adjunct modifiers such as adjectives and adverbials are associated with a logical expression containing the appropriate intrinsic literal coupled with an extrinsic literal that specifies the thematic relation between the modifier and its head. For example, the semantics of *"indah"* (beautiful) is $\lambda A \lambda T \;\; property(A, indah) \wedge attrib(T, A)$.

The lexical semantics of prepositions and words which coordinate and/or subordinate other clauses is simply the appropriate extrinsic literal which specifies the relation between the prepositional phrase and its head or the clauses being coordinated. For example, the semantics of *"karena"* (because) is $\lambda X \lambda Y \;\; cause(X, Y)$.

Question words, i.e. *wh* words in Indonesian, e.g. *"apa"* (what), *"siapa"* (who), *"mana"* (mana), are associated with a logical expression that contains two literals. The first is the appropriate literal which would typically be associated with the answer, but instead of specifying the domain concept as the second argument, it is given a variable *Ans*. The second literal is a special $query(Ans)$ literal that indicates a question that is to be processed by the question-answering module (see Section 4 for details). For example the lexical semantics of *"siapa"* is $\lambda X \;\; person(X, Ans) \wedge query(Ans)$.

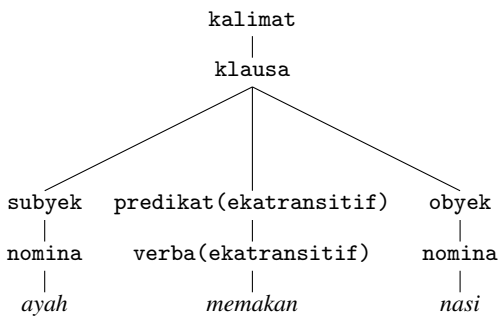
Finally, there are several special cases of lexical semantics where morphological processes introduce literals. For example, the suffix *"-nya"* amounts to a possessive pronoun, requiring the addition of the literals $person(O, owner)$ and $possess(O, X)$ to the lexical semantics. For example, we assume that the lexical semantics for the word *"bukunya"* (his/her book) is $\lambda X \;\; object(X, buku) \wedge person(O, owner) \wedge possess(O, X)$.

3.3 Semantic attachment rules

Montague's principle of compositionality of semantics states that the meaning of a complex expression is determined by the meanings of its parts, and the way in which those parts are combined. However, it says nothing about how exactly these meanings can be constructed. For this, we turn to the well-known rule-to-rule hypothesis (Bach, 1976), which states that semantic interpretation rules are defined on the syntactic rules and structures. As a result, we develop *semantic attachment rules* for each syntactic rule in our grammar.

These semantic attachment rules define how the lexical semantics of the constituent words are combined, and in particular how the correct predicate-argument structure is specified. The most common approach is to use *lambda calculus* notation, where predicate-argument structure is controlled through the operation of β-reduction. See Jurafsky and Martin (2000) for a clear discussion of this approach (note that they call the process lambda-reduction).

To see an example of the semantic attachment rules and how they are combined, observe the following example, which constructs the semantic representation of the simple declarative sentence *"Ayah memakan nasi"* (father eats rice).



The sentence is parsed by our grammar (Sect. 2.1) as follows: a `kalimat` (sentence) can consist of a single `klausa` (clause), which in turn expands to a transitive verbal predication. The `subyek` and `obyek` categories simply consist of a single `nomina` lexeme, whereas a transitive `predikat` consist of a single transitive `verba` lexeme (note the `ekatransitif` subcategorization value). Rules (1)-(4) below show the required syn-

tax rules and corresponding semantic attachment rules, whereas rules (5)-(7) show the lexical semantics entries (see Section 3.2 for discussion of these values):

1. `klausa -> subyek, predikat(ekatransitif), obyek`
   $\lambda M\ \ predikat.sem(M)(K)(N) \wedge subyek.sem(K) \wedge obyek.sem(N)$

2. `subyek -> nomina`
   $\lambda X\ \ nomina.sem(X)$

3. `predikat(ekatransitif) -> verba(ekatransitif)`
   $\lambda Y\ \ verba.sem(Y)$

4. `obyek -> nomina`
   $\lambda Z\ \ nomina.sem(Z)$

5. `nomina -> [ayah]`
   $\lambda D\ \ person(D, ayah)$

6. `verba(ekatransitif) -> [memakan]`
   $\lambda E \lambda A \lambda P\ \ event(E, makan) \wedge agent(E, A) \wedge patient(E, P)$

7. `nomina -> [nasi]`
   $\lambda S\ \ object(S, nasi)$

The *.sem* operator indicates the logical expression of the indicated syntactic category. The β-reduction proceeds as follows:

1. Lexical semantics are copied over to the `subyek`, `predikat`, and `obyek` categories:

   (2) & (5): `subyek.sem=`
   $\lambda X\ (\lambda D\ person(D, ayah))(X)$
   reduces to
   $\lambda X\ person(X, ayah)$

   (3) & (6): `predikat.sem=`
   $\lambda Y\ (\lambda E \lambda A \lambda P\ \ event(E, makan) \wedge agent(E, A) \wedge patient(E, P))(Y)$
   reduces to
   $\lambda Y \lambda A \lambda P\ \ event(Y, makan) \wedge agent(Y, A) \wedge patient(Y, P)$

   (4) & (7): `obyek.sem=`
   $\lambda Z\ (\lambda S\ object(S, nasi))(Z)$
   reduces to
   $\lambda Z\ object(Z, nasi)$

2. At the `clause` rule, the semantics of `subyek`, `predikat`, and `obyek` are substituted and reduced:

   (1),(2),(3) & (4): `klausa.sem=`
   $\lambda M\ \ (\lambda Y \lambda A \lambda P\ \ event(Y, makan) \wedge agent(Y, A) \wedge patient(Y, P))(M)(K)(N) \wedge (\lambda X\ person(X, ayah))(K) \wedge (\lambda Z\ object(Z, nasi))(N)$
   reduces to
   $\lambda M\ \ event(M, makan) \wedge agent(M, K) \wedge patient(M, N) \wedge person(K, ayah) \wedge object(N, nasi)$

The semantic representation of `kalimat`, the sentence, is simply the semantics of the `clause` as shown above.

4 Initial implementation

Our initial implementation is written in Prolog, to exploit its facilities of unification and search. However, a preliminary step was required to translate the rules from Joice (2002), which were written in the PC-PATR format, into a form that could be used by our implementation. To achieve this, we wrote a Perl script that processed a PC-PATR rule and wrote out the equivalent DCG rules. In particular, since PC-PATR rules allow for disjunction and optional constituents, a single PC-PATR rule was translated into a set of DCG rules that enumerated all possible configurations. Due to the very inelegant definition of rules (see Section 2.1), our initial attempt at translating the 61 PC-PATR rules resulted in over seven thousand DCG rules, which contained massive redundancy. At the expense of reduced coverage, we then decided to prune the grammar down to an elementary form by omitting all optional constructions. Further elimination of redundant rules provided us with a "skeletal" grammar of 421 rules.

Each of the syntactic rules from this grammar was augmented with semantic attachment rules (see Section 3.3), and we also constructed a handcrafted lexicon where words were associated with lexical semantics as discussed in Section 3.2.

A prototype Prolog parser with semantic representation building, using the associated attachment rules, was developed to handle our resources. Testing on a set of 496 sentences obtained from a national newspaper revealed that 66 sentences were successfully parsed. This low percentage of coverage was due to the aforementioned grammar pruning and limited lexicon. Manual observation of the parsed sentences showed that the correct semantic representations were being constructed.

The next step was to develop a question-answering module. In general, we assert new facts to the knowledge base by "telling" it Indonesian declarative sentences. This can be repeated for as many sentences as necessary. Finally, we issue a query to the knowledge base by "asking" it an Indonesian interrogative sentence.

A typical evaluation of a question answering system requires a standardized set of documents and a series of question whose answers are to be found within the documents, such as those used by the TREC and CLEF challenges. Unfortunately no such resource currently exists for Indonesian, so we performed a very limited testing of the sys-

tem's ability to extract answers from some artificially created questions. This testing confirmed that our system can currently account for some lexical and syntactic variation, and can correctly provide multiple answers to a question, if warranted by the input texts.

For example, say we assert the following sentence:

Presiden	*mengumumkan*	*susunan*	*kabinet*
The president	announced	roster	cabinet
baru	*kepada*	*wartawan*	*di*
new	to	reporters	at
istana.			
palace			

("The president announced the new cabinet roster to reporters at the palace")

using the following Prolog statement:

```
tell_KB([presiden,mengumumkan,susunan,
         kabinet,baru,kepada,wartawan,
         di,istana]).
```

The knowledge base will assert all the individual literals contained in the resulting semantic representation as follows:

```
event(x1, umum).
agent(x1, x2).
patient(x1, x3).
object(x2, presiden).
object(x3, susunan).
object(x4, kabinet).
nn(x3, x4).
adjective(x5, baru).
attrib(x4, x5).
object(x6, wartawan).
purpose(x1, x6).
object(x7, istana).
di(x1, x7).
```

Note that the predicate argument structure is now represented through atomic Prolog terms `x1`, `x2`, etc. This is because the scope of Prolog variables is only local to the clause it appears in.

Finally, we can query the database the following question:

Siapa	*mengumumkan*	*susunan*	*kabinet*	*baru*	?
Who	announced	roster	cabinet	new	?

("Who announced the new cabinet roster?")

using the following Prolog query:

```
ask_KB([siapa,mengumumkan,susunan,
        kabinet,baru], Ans).
```

This will construct a Prolog `query/1` rule by constructing the semantic representation of the question, and extracting the *query* literal contained by the *wh*-lexeme (see Section 3.2) from the expression and making it the head of the rule.

The resulting rule is as follows:

```
query(Ans) :- event(A, umum),
              agent(A, B),
              patient(A, D),
              object(B, Ans),
              object(D, susunan),
              object(E, kabinet),
              adjective(F, baru),
              attrib(E, F),
              nn(D, E).
```

Executing this query will unify the query variable with the correct result: `presiden`.

The following sentence is a slightly more complex one:

Wartawan	*melaporkan*	*bahwa*	*harga*	*susu*
Journalists	report	that	price	milk
meningkat	*karena*	*pasokan*	*susu*	*impor*
increase	because	supply	milk	import
berkurang	*di*	*Indonesia.*		
decrease	in	Indonesia		

("Journalists have reported an increase in the price of milk due to a decrease in imported milk supplies in Indonesia.")

Given this sentence, our semantic analyser constructs the following semantic representation:

```
event(x1, lapor).
agent(x1, x2).
patient(x1, x3).
profession(x2, wartawan).
complement(x3, x4).
event(x4, naik).
agent(x4, x5).
object(x5, harga).
object(x6, susu).
nn(x5, x6).
event(x7, kurang).
agent(x7, x8).
object(x8, pasokan).
object(x9, susu).
object(x10, impor).
nn(x9, x10).
nn(x8, x9).
location(x11, indonesia).
di(x7, x11).
cause(x4, x7).
```

This enables us to query the database with, for example, the following question:

Pasokan	*apa*	*yang*	*berkurang*	?
Supplies	what	which	decreased	?

("What supplies have decreased?")

```
ask_KB([pasokan,apa_yang,
        berkurang,?], Ans).
```

The resulting rule is as follows:

```
query(Ans) :- event(B, kurang),
              agent(B, C),
              location(C, pasokan),
              object(D, Ans),
              nn(C, D).
```

Executing this query will unify the query variable *Ans* with the correct result: `susu`.

5 Further work & summary

We believe that our implementation forms an important basis for the construction of a question-answering system for Bahasa Indonesia based on deep linguistic analysis. However, several issues must be addressed before this can be realised:

1. The source grammar we started with, Joice (2002), was linguistically unelegant. For our prototype, we extracted a "skeletal" grammar that was a suitable size to work with. Unfortunately, coverage of the grammar was significantly compromised. The next step is to gradually reintroduce various syntactic constructions that were omitted as a result.

2. Our implementation is essentially a proof-of-concept system. It would certainly be naive to expect we could handle a corpus in the order of millions of words by asserting them as logical facts in a Prolog knowledge base. We plan to embed our system within a more conventional statistical QA system, where it would work with a small passage of texts that had previously been filtered through standard IR techniques.

3. Our account of lexical semantics is very naive in that we arbitrarily choose some synonym as a token representing a domain concept. As work on an Indonesian version of WordNet progresses, we hope to gradually update our lexicon.

The system described in this paper can be downloaded from our Information Retrieval Lab webpage at `http://ir.cs.ui.ac.id`.

References

Hiyan Alshawi and Jan van Eijck. 1989. Logical forms in the core language engine. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 25–32. Association for Computational Linguistics, Vancouver, Canada.

Hasan Alwi, Soenjono Dardjowidjojo, Hans Lapoliwa, and Anton Moeliono. 1998. *Tata Bahasa Baku Bahasa Indonesia*. Balai Pustaka, Jakarta, Indonesia, third edition.

Emonn Bach. 1976. An extension of classical transformational grammar. In *Problems of Linguistic Metatheory: Proceedings of the 1976 Conference*, pages 183–224. Michigan, USA.

Johan Bos and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*. Venice, Italy.

Benjamin Van Durme, Yifen Huang, Anna Kupść, and Eric Nyberg. 2003. Towards light semantic processing for question answering. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, pages 54–61. Association for Computational Linguistics, Edmonton, Canada.

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

Jerry Hobbs. 1985. Ontological promiscuity. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 61–69. The Association for Computational Linguistics, Chicago, USA.

Joice. 2002. Pengembangan lanjut pengurai struktur kalimat bahasa indonesia yang menggunakan constraint-based formalism. Undergraduate thesis, Faculty of Computer Science, University of Indonesia.

Daniel S. Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall.

Stephen McConnel. 1995. *PC-PATR Reference Manual*. Summer Institute for Linguistics. `http://www.sil.org/pcpatr/manual/pcpatr.html`.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic struc-

tures. In *Proceedings of the 20th international conference on Computational Linguistics*.

Ayu Purwarianti, Masatoshi Tsuchiya, and Seiichi Nakagawa. 2007. A machine learning approach for indonesian question answering system. In Vladan Devedzic, editor, *Proceedings of the International Conference on Artificial Intelligence and Applications (AIA 2007)*. IASTED, Innsbruck, Austria.

Stuart Shieber. 1986. An introduction to unification-based approaches to grammar. CSLI Lecture Notes 4, Center for the Study of Language and Information, Stanford, USA.

Sri Hartati Wijono, Indra Budi, Lily Fitria, and Mirna Adriani. 2006. Finding answers to indonesian questions from english documents. In *Working Notes of the Workshop in Cross-Language Evaluation Forum (CLEF)*. Alicante.

Query Recommendation from Search Engine Query Logs Using Query Enrichment

Shisanu Tongchim and **Virach Sornlertlamvanich**
Thai Computational Linguistics Laboratory
NICT Asia Research Center
112 Paholyothin Road, Klong Luang
Pathumthani, Thailand
{shisanu,virach}@tcllab.org

Hitoshi Isahara
Computational Linguistics Group
NICT
3-5, Hikari-dai, Seika-cho
Soraku-gun, Kyoto, 619-0289, Japan
isahara@nict.go.jp

Abstract

This article presents a technique to find semantically related queries from the search engine query log. The proposed technique is applied to the query recommendation task. The experiment is carried out on the log of user queries written in Thai. We first discuss the nature of Thai queries that makes the problem of finding related queries much more challenging. Not only are the queries short and ambiguous, the ambiguity of word boundary and ill-written queries also make the fundamental process of Thai text processing like word segmentation problematical. To cope with these problems, we propose the use of results from search engines to expand and enrich the meanings of queries. The experimental results show that the proposed technique is quite robust although the errors of word segmentation inevitably occur.

1 Introduction

Content suggestion becomes an ordinary tool for enhancing user interfaces and improving usability. In e-Commerce websites, content suggestion has been used for introducing related items that may interest buyers. For web search services, content suggestion has been used for recommending related queries issued by other users or some related advertisements to the user queries. Users may benefit from query recommendation in refining their original queries. The suggested queries are normally adopted from the log of previously issued queries. However, finding related queries from the query log of search engines is not a trivial task. Queries are generally short and ambiguous. Many of them contain typos or ill-written strings. The use of textual content of queries alone is insufficient in finding the similarity among queries in many cases. Due to the limited information in query keywords, some studies utilize human activities recorded by search engines. These studies try to extract the information of user sessions from the search engine logs. Some proposed techniques (Chan et al., 2004; Baeza-Yates et al., 2004) rely on the information about user queries and the documents selected by users for the corresponding queries (called *clickthroughs*). The assumption of these techniques is that two queries are similar if both queries lead to the same or related documents. Some proposed techniques (Fonseca et al., 2004, 2003) try to identify the set of queries issued by a user within the same period. The intuition behind these techniques is that a user submits several related queries during searching for information.

The techniques mentioned earlier assume that the search engine log recording user activities is available. They can be considered as passive methods. That is, the input queries have to be found in the search engine log and the information about these queries (e.g., clickthroughs) have to be known. In this paper, we propose a technique that can be regarded as an active method. The proposed technique can be used in the case that only a list of queries is available as the input data. The input queries are unnecessary to be found in the query log, although the related queries suggested by our system are limited to the known queries from the query log. To resolve the ambiguity of short and ill-written query strings, the queries are submitted to a number of search engines and the results of search engines are used as features for determining keyword similarity. This process is called *query enrichment*. The experimental results show that the proposed technique helps in improving the performance when comparing with the use of words from queries alone.

The rest of this paper is organized as follows: Section 2 presents some related studies. Section 3 discusses the nature of Thai queries that makes

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 281–288,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

the problem of finding related queries much more challenging. Section 4 describes the proposed technique. Section 5 presents the experimental results. Finally, Section 6 concludes our work.

2 Related Work

Since the similarity measurement of queries is the fundamental part of query clustering, the research areas on query similarity measurement and query clustering are closely related. Fu et al. (2004) categorized the techniques for finding query similarity and clustering search queries according to features used as follows:

1. *Content-based approaches* : These approaches utilize the words from queries for finding similarity. The similarity measurement can be done by using the number of shared terms between two queries.

2. *Feedback-based approaches* : These approached use feedback from users, e.g. the list of selected results from users, for analyzing query similarity. The assumption is that two queries are related if the documents selected by users for both queries have a high degree of overlap.

3. *Result-based approaches* : These approaches use the documents returned for the queries for finding the similarity. This can be done by using text found in the returned documents to create term frequency vectors for making comparisons, or using the overlap of result URLs instead of the textual content.

Fu et al. (2004) conducted an experiment for comparing two content-based similarity approaches and a result-based approach. The result-based approach in their experiment was based on the overlap of result URLs, while two content-based approaches were based on the overlap of terms in queries. The results showed that the result-based approach performed well in terms of precision, but not for recall. The content-based approaches had higher recall values at the expense of precision.

Since search queries are typically short, the similarity measurement based solely upon query words as the content-based approaches is rarely found in practice. Most studies in finding the similarity of queries use some additional information, rather than utilizing only the extracted words from query strings. Baeza-Yates et al. (2004) used a set of clicked URLs (called *clickthroughs*) as a feature vector for each query. The extracted words from the clicked URLs were used to determine the query similarity. This approach assumes that a log containing the information about clicked URLs is available. Chan et al. (2004) proposed a hybrid between the use of query words and the information of clicked URLs. Wen and Zhang (2003) evaluated the similarity function based on query keyword, the similarity function using clickthroughs and the hybrid functions. The results suggested that the hybrid functions obtained best overall performance. Fonseca et al. (2004, 2003) assumed that a set of queries submitted to a search engine by one user during a short period is a set of related queries. They proposed the use of association rules to find related queries. The same concept was used by Zhang and Nasraoui (2006) for the query recommendation task.

In this paper, we use a technique called query enrichment to expand the meanings of input queries. This technique has been used by some researchers to resolve the ambiguity of queries. It works by submitting queries to search engines, and then uses the returned results to produce an enrichment of the queries. Query enrichment was used by the team who won ACM KDDCUP in 2005 (Shen et al., 2006). The task was to classify given queries to predefined categories. This work used search engines to find category information. They first submitted queries to directory search in order to obtain related pages and the page's categories. The authors claimed that these categories labeled by humans are useful in categorizing search queries. If no result can be obtained from the directory search, queries are sent to ordinary web search. In our case, however, the web directories are less developed in the case of Thai websites. Thus, we only use the returned results from ordinary web search.

3 Nature of Thai Queries

The list of Thai queries used in this study is provided by Truehits[1]. Truehits is the largest web statistics collector in Thailand. The usage statistics collection has been done by providing a small script to members. Every member places this script on their web pages. Each time these web pages are viewed, some information has been col-

[1] truehits.net

lected and sent to Truehits. Various kinds of information have been collected, for example, web browser vendor, screen resolution, operating system, referrer information, etc. By analyzing the referrer data, Truehits can identify what pages users were visiting or accessing immediately before coming to the current page. Thus, Truehits can keep track of what search engines and keywords users were using to find the websites of Truehits members.

Currently, Truehits has recorded that there are more than 3 million search engine usages per day (as of March 11, 2007). According to this number, about 480,000 keywords have been used in each day. These numbers reflect the amount of usage recorded by Truehits 's members. Thus, the real numbers should be larger than these values. Truehits also records keywords used by users to find websites. This list of keywords is obtained from the analysis of the referrer data. Thus, they will be queries conforming to the following criteria:

- The recorded keywords are extracted directly when users follow the links provided by search engines to Truehits 's members.
- The results of search engines that are not members of Truehits will not be recorded, even though users follow the result links.
- By using these keywords, at least one result will be found. Keywords without any results found cannot be recorded.
- They are recorded when users follow the links from the result pages of search engines. In case that some keywords are queried to search engines and users do not follow the links in search results, these keywords cannot be recorded.

These criteria are also applied to other statistics about search engines. Some results have to be found by search engines first and they seem to be relevant in order to attract users to follow the result links. Therefore, we expect that the number of keywords in real usage should be larger than this. Improper keywords without any returned results from search engines will not be recorded. Likewise, search engines that return uninteresting results may not be recorded since users may not follow the provided links to websites. Although only successful keywords have been recorded, the recorded keywords still provide some useful information and insights about how users use search engines.

Thai language can be regarded as a non-segmenting language (like Chinese, Korea and Japanese). Thai has no explicit word nor sentence boundary. This feature has an important influence on the keyword formation and the results obtained. When analyzing the log of keywords used by users, there are two main types of queries. The first one is like English queries. Each query is composed of one or several individual words separated with spaces. That is, users manually select important words as queries. The second query type is composed of one or few short phrases, or sometime (in a few cases) a short sentence. In general, they look like short phrases without manual word segmentation.

Intuitively, the use of short phrases as queries is more specific, but may lack in the number of returned results. Most search engines try to find exact matches for the input Thai query strings. The longer the queries are, the fewer number of results will be obtained. For example, if we submit a query string like "วิธีทำกล้วยตาก" (วิธี/how to; ทำ/make; กล้วยตาก/Kluai-Tak) to Google, only 4 results can be found. If we separate this query into two words as "วิธีทำ" and "กล้วยตาก", and submit this query to Google again, about 300 pages can be found.

Like the ordinary list of keywords submitted to search engines, the list of Thai keywords from Truehits has diverse characteristics. Many of them are short and ambiguous. Many queries contain typos or ill-written strings. Many queries are clean and specific, like names of organization or persons. To find the similarity of these queries, each query is transformed into a bag of words. The word segmentation that is a fundamental process for Thai text processing have to be performed. The correctness of the word segmentation clearly influences the results of later steps in finding the similarity of queries. Based on the query log analysis, there are several issues that possibly degrade the performance of word segmentation and may affect the query similarity calculation.

- **Unknown word problem**

  New words regularly appear throughout the Web. Since many word segmentation algorithms rely on dictionaries, the accuracy and completeness of dictionaries play an impor-

tant role in the correctness of word segmentation. For example, the query "ก้านกล้วย" (Kan Kluai) which is the name of new Thai animation will separated into two words, "ก้าน" (limb) and "กล้วย" (banana), by the word segmentation. When comparing this query with others, it may be confused by some queries containing the word "กล้วย" (banana).

- **Improper query usage**

  Improper queries are typically found in the query log. They contains typos, ill-written strings or informal language. For example, a query found in the query log is "กรมอุตุนิยมวิทยา" which is the "Thai Meteorological Department". This query can be considered as two words: "กรม" (department) and "อุตุนิยมวิทยา" (meteorology). Since this word represents a unique entity, it may be recognized as an indivisible unit. We have found that there are some queries that resemble to this word, but they are ill-written. For example, we found at least three queries that can be considered to refer to this word: "กรมอุตุ", "กรมอุตุนิยม" and "กรมอุตุวิทยา". The use of these keywords usually leads to the websites that have improper forms of the word "กรมอุตุนิยมวิทยา", rather than the website of the Thai Meteorological Department.

  When applying the word segmentation to these queries, the results may be unexpected. For example, the query "กรมอุตุนิยม" will be separated into three words: "กรม" (department), "อุตุ" (comfort) and "นิยม" (favour). Thus, it is difficult when trying to relate this query to "กรมอุตุนิยมวิทยา".

- **Incorrect or inconsistency transliteration**

  There are many transliterated words in the list of queries. They are foreign words written by using Thai characters. One problem arises when each user has a different way for transliteration. For example, the word "Internet" has been transliterated into several forms: "อินเตอร์เน็ต", "อินเทอร์เน็ต" and "อินเตอร์เน็ท". This issue affects the similarity measurement among queries that have different forms of transliteration. Moreover, the dictionary or lexicon used by the word segmentation algorithm may not have all forms of transliterated words. In this case, the unknown word problem may occur.

In addition to the short and ambiguous queries typically found in the most search engine query logs, Thai queries have some characteristics that are prone to the inherent errors of word segmentation and the query similarity measurement. The issues mentioned earlier are likely to have strong effect when the query similarity measurement is based solely on the textual information in queries. In the next section, the method to expand the meanings of queries is presented. The proposed method utilizes the results of several search engines as additional features to measure the query similarity. This technique makes the query comparison less sensitive to the problems found in queries.

4 Query Enrichment

We use the information from the Web to resolve and expand the meanings of queries. The web can be regarded as large corpora and search engines are tools to find meaningful information from them. All search engines try to find the results that are most likely to be relevant to the input query. The returned results from search engines can be viewed as a valuable source of information for resolving the meanings of queries.

The process of query enrichment is as follows:

1. We submit each query to a number of public web search engines. All queries are not preprocessed. Due to the process of query recording mentioned before, we expect that we are able to retrieve the results for most queries. Few queries make search engines difficult to find any relevant results. By using several search engines, this ensures that some results can be found for such queries.

2. Typically, the returned results from search engines are shown as the titles of web pages, short descriptions and hyperlinks. These short descriptions, sometime called snippets, are adopted from the web documents. Our assumption is that snippets shown by search engines are relevant to the given queries. We use the textual information from the titles and snippets as representatives of the parts of web documents relating to the queries. The titles and snippets are parsed from the returned results, and merged into a single document. The word segmentation is applied to extract words from the titles and snippets. The ex-

tracted words from the query string are also merged into the list of extracted words from the titles and snippets. Stopwords and special symbols are filtered out. Stemming is not applied since Thai language does not have the concept of inflection.

3. We filter out the words that have low frequency. By using a simple rule, the words that appear less than 1 time in 10 returned results are filtered out. The extracted words are then weighted by using *tf*idf*.

Some examples of query enrichment are shown in Table 1. In the first query, เกมส์ออนไลน์ is composed of two words: เกมส์ (games) and ออนไลน์ (online). The word "เกมส์" is the transliterated word of "games". However, there is another form of this word like "เกม" and both forms can be used interchangeably. From the list of the top 10 words, the use of query enrichment helps in correlating the query words with the other relevant words, e.g. "เกม" and also in English "game". In the second query, the words "เลี้ยง" and "เพาะเลี้ยง" have a similar meaning ("breeding") in this context and can be used interchangeably. This example shows that the word "เพาะเลี้ยง" can be connected to this query although it does not appear in the query.

5 Experimental results

We evaluate the proposed method in the query recommendation task. Given a set of 9000 queries adopted from Truehits, we select 60 queries as the main topics. These main topics are verified by hand that all of them have at least one related query. Each main topic is specific enough in order to avoid an excessive number of related queries. The task is to find the top 30 queries that are likely to be relevant to each given main topic. The evaluation of recommended queries is carried out by a team of three judges. The relevance judgement is binary. Each recommended query is judged whether it is relevant to the main topic or not. We adopt the mean average precision (MAP) as the evaluation measure. MAP is extensively used in IR research to illustrate the overall effectiveness of a system (Voorhees and Harman, 2005). MAP ranges from 0 to 1. A higher score is better.

To assess the benefit of using multiple search engines for query enrichment, we conduct some experiments by using one and three search engines. In the experiment using three search engines, we submit the queries to Google[2], SiamGURU[3] and AlltheWeb[4]. These engines are the top three search providers on Thai queries (Tongchim et al., 2007, 2006). In the experiment using one search engine, the queries are submitted to Google. The top 20 results from each search engine are used. The word segmentation is based on the maximal matching method.

To measure the similarity of two queries, a term-weight vector for each query is made using query enrichment. Given two queries a and b with the number of extracted terms m and n respectively. Let l be the number of shared terms between both queries. The cosine similarity is used to compute the similarity score between a pair of queries as follows:

$$sim(a,b) = \frac{\sum_{i=1}^{l} cw_i(a) \times cw_i(b)}{\sqrt{\sum_{i=1}^{m} w_i^2(a) \times \sum_{i=1}^{n} w_i^2(b)}} \quad (1)$$

where $w_i(a)$ and $w_i(b)$ are the weights of the i-th terms in the query a and b respectively, and $cw_i(a)$ and $cw_i(b)$ are the weights of the i-th common term in the query a and b respectively.

Each term is weighted by using *tf*idf*. We also conduct an experiment using query words without query enrichment. This test is used as a baseline for assessing the benefit of the proposed method. However, using *tf*idf* for the query words is impractical. Most query words appear one time only in each query. Thus, *tf* for the query words makes no sense. Some studies suggest the use of the search frequency instead of the term frequency (Zhang and Nasraoui, 2006). However, Truehits does not provide the information about the search frequency. Thus, we use *idf* to weight the extracted terms in the experiment on the query words.

In the preliminary experiment, we observe that the use of query words without query enrichment tends to produce the results with high precision, but low recall. The use of query enrichment is less precise in terms of precision, but it helps in improving recall. Thus, we propose a hybrid method by combining the similarity scores from two systems as a way to improve recall while maintaining precision. A simple combination is used as follows:

[2] www.google.com
[3] www.siamguru.com
[4] www.alltheweb.com

Table 1: Some examples of query enrichment

Query	Top 10 words according to tf*idf
เกมส์ออนไลน์ (Online games)	1. เกมส์ (games) 2. ออนไลน์ (online) 3. เกม (game) 4. เล่น (play) 5. game 6. ร้าน (shop) 7. แต่งตัว (dress) 8. มั่ง (as well) 9. ฟรี (free) 10. ต่อสู้ (fighting)
การเลี้ยงปลาทะเล (Sea fish breeding)	1. ปลา (fish) 2. เลี้ยง (breeding) 3. ทะเล (sea) 4. สวยงาม (beautiful) 5. ทะเลใน (inland sea) 6. กระชัง (floating basket for containing live fish) 7. การ (job) 8. ตู้ปลา (aquarium) 9. ตู้ (cabinet – In this context, it refers to ตู้ปลา/aquarium.) 10. เพาะเลี้ยง (breeding)

$$sim_{hybrid} = \alpha \times sim_{enrich} + \beta \times sim_{query} \qquad (2)$$

where sim_{enrich} and sim_{query} are the cosine similarity scores with and without query enrichment respectively.

In the hybrid method, sim_{enrich} is based on the results from three search engines. To select the best parameter setting, we fine-tune the parameters by using a small set of 6 topics apart from the topics used in the main evaluation. Without much effort, we found that $\alpha = 0.9, \beta = 0.1$ yields the best performance.

The experimental results are shown in Table 2. The first system relies on the words from queries without using query enrichment. The second and third systems use query enrichment based on the results from 1 and 3 search engines respectively. The last system is the hybrid between the first and third systems. The results show that the hybrid method outperforms all other systems in the experiment. The use of 3 search engines in query enrichment clearly shows benefit over the use of just 1 engine. Query enrichment with 1 engine produces slightly better performance than the use of query words alone. To obtain accurate comparison, we use *repeated-measures ANOVA* to compare the performance of these methods. The use of repeated-measures ANOVA assumes that the assumption of sphericity is not violated. Mauchly's test is used to test whether the sphericity assumption is violated or not. The results of the test indicate that this assumption is violated ($\chi^2(5) = 102.377, p < .001$). Thus, degrees of freedom are corrected using Greenhouse-Geisser estimates of sphericity ($\varepsilon = .684$). The results of repeated-measures ANOVA show that there are significant differences among the performance of these methods, $F(2.052, 121.040) = 8.624, p < .001$. The pairwise comparisons using the Bonferroni correction ($p < .05$) are then performed. The pairwise comparisons reveal that the hybrid and third methods statistically outperform the first and second methods, while there is no statistically difference between the hybrid and the third methods.

From the results, query enrichment helps in improving the performance when it utilizes the results from multiple search engines, rather than just one engine. Moreover, query enrichment should be used in combination with the similarity measure using the query words as the hybrid method. To illustrate how the suggested queries look like, the first 10 related queries for some main topics found by the hybrid approach are shown in Table 3. In order to make the comparison between words in these queries easier, the slash symbol shows the word boundaries that are not separated by spaces. In the first main query, there are several ways to express the phrase "to grow mushroom" in Thai, i.e. การเพาะ, การปลูก and การทำ. All of these words can be found to relate to the main topic. In the second example, it is used to illustrate how the proposed technique performs when the given main topic has improper query usage. The given main topic is "กรมอุตุ" that refers to "กรมอุตุนิยมวิทยา" (Thai Meteorological Department). The word segmentation algorithm that we use has "กรมอุตุนิยมวิทยา" in its dictionary. Thus, this string is considered as an indivisible word. In contrast, the main topic "กรมอุตุ" is considered as two words, i.e. "กรม" (department) and "อุตุ" (comfort), by the word segmentation algorithm. By relying solely on the extracted words from queries, it is impossible to find any correlation between both queries. This is also applied to other ill-written queries like กรมอุตุวิทยา, กรมอุตุนิยม. From the table, the query "กรมอุตุนิยมวิทยา" is found in the sev-

Table 2: MAP for the query recommendation task

	Method	MAP
1	Query Words	0.560
2	Query Enrichment (1 search engine)	0.597
3	Query Enrichment (3 search engines)	0.693
4	Hybrid	0.711

enth and tenth suggested queries[5]. This is an example of how query enrichment helps in correlating the queries that cannot be done by using the words from query strings. Note that several unrelated queries receive high ranks since they contain the word "กรม" that matches with the same word in the main topic. Many of these unrelated queries are ill-written and they are separated into several words like the main topic. The extracted word "กรม" found in these queries is the main source of false similarity.

6 Conclusions

We have proposed the use of search engine results to improve the similarity measurement among queries. This technique is referred to as query enrichment. We use the query recommendation task to examine the feasibility of the proposed technique. The experimental results show that the hybrid between the use of query words and query enrichment statistically outperforms the use of query words alone. We also provide some examples of how the proposed technique helps in resolving the ambiguity of improper query usage.

Acknowledgements

The authors would like to thank Mr. Kergrit Robkop who implemented a program for fetching results from search engines.

References

Ricardo A. Baeza-Yates, Carlos A. Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In Wolfgang Lindner, Marco Mesiti, Can Türker, Yannis Tzitzikas, and Athena Vakali, editors, *EDBT Workshops*, volume 3268 of *Lecture Notes in Computer Science*, pages 588–596. Springer.

Wing Shun Chan, Wai Ting Leung, and Dik Lun Lee. 2004. Clustering search engine query log containing noisy clickthroughs. In *SAINT*, pages 305–308. IEEE Computer Society.

Bruno M. Fonseca, Paulo Braz Golgher, Edleno Silva de Moura, Bruno Pôssas, and Nivio Ziviani. 2004. Discovering search engine related queries using association rules. *J. Web Eng.*, 2(4):215–227.

Bruno M. Fonseca, Paulo Braz Golgher, Edleno Silva de Moura, and Nivio Ziviani. 2003. Using association rules to discover search engines related queries. In *LA-WEB*, pages 66–71. IEEE Computer Society.

L. Fu, D.H. Goh, and S.S.B. Foo. 2004. The effect of similarity measures on the quality of query clusters. *Journal of Information Science*, 30(5):396–407.

Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. 2006. Query enrichment for web-query classification. *ACM Trans. Inf. Syst.*, 24(3):320–352.

Shisanu Tongchim, Virach Sornlertlamvanich, and Hitoshi Isahara. 2006. Measuring the effectiveness of public search engines on thai queries. In *Proceedings of The Fifth IASTED International Conference on communications, internet, and information technology*.

Shisanu Tongchim, Virach Sornlertlamvanich, and Hitoshi Isahara. 2007. Examining the feasibility of metasearch based on results of human judgements on thai queries. In *Proceedings of The 2007 IEEE International Symposium on Data Mining and Information Retrieval*.

Ellen M. Voorhees and Donna K. Harman. 2005. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press.

Ji-Rong Wen and HongJiang Zhang. 2003. Query clustering in the web context. In Weili Wu, Hui Xiong, and Shashi Shekhar, editors, *Clustering and Information Retrieval*, pages 195–226. Kluwer.

Zhiyong Zhang and Olfa Nasraoui. 2006. Mining search engine query logs for query recommendation. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *WWW*, pages 1039–1040. ACM.

[5]The tenth query differs from the seventh query that it has a full stop.

Table 3: Some examples of query recommendation

Main query	Suggested queries
การ/เพาะ/เห็ดฟาง/ใน/โรงเรือน (How to grow straw mushroom in the cultivation building)	การ/เพาะ/เห็ดฟาง (How to grow straw mushroom) การ/ทำ/เห็ดฟาง (How to grow straw mushroom) การ/เพาะ/เห็ด (How to grow mushroom) การ/ปลูก/เห็ดฟาง (How to grow straw mushroom) การ/เพาะ/เห็ดหอม (How to grow Shiitake mushroom) การ/เพาะ/เห็ดนางฟ้า (How to grow Sajor-caju mushroom) กอง/ฟาง (Pile of straw) การ/เพาะ/เห็ดโคนญี่ปุ่น (How to grow Yanagi Mutsutake mushroom) การ/ปลูก/เห็ด (How to grow mushroom) การ/เพาะ/กาย (Bodybuilding)
กรม/อุตุ (Improper keyword usage that refers to กรมอุตุนิยมวิทยา – Thai Meteorological Department)	กรม/อุตุ/วิทยา (like กรมอุตุ) กรม/อุตุ/นิยม (like กรมอุตุ) กรม (Department) กรม/สิ่งแวดล้อม (Department of environment) กรม/ปกครอง (Department of Administration) กรม/ส่งเสริม/การลงทุน (Investment Promotion Department) กรมอุตุนิยมวิทยา (Thai Meteorological Department) กรม/ปกครอง/ส่วนท้องถิ่น (Department of local administration) กรม/พัฒนา/ท้องถิ่น (Department of local development) กรมอุตุนิยมวิทยา. (Thai Meteorological Department.)

Note: The symbol “/” is used to illustrate the word boundaries. It does not exist in the real Thai text.

Automagically Inferring the Source Words of Lexical Blends

Paul Cook
Department of Computer Science
University of Toronto
Toronto, Canada
pcook@cs.toronto.edu

Suzanne Stevenson
Department of Computer Science
University of Toronto
Toronto, Canada
suzanne@cs.toronto.edu

Abstract

Lexical blending is a highly productive and frequent process by which new words enter a language. A blend is formed when two or more source words are combined, with at least one them shortened, as in *brunch* (*breakfast+lunch*). We use linguistic and cognitive aspects of this process to motivate a computational treatment of neologisms formed by blending. We propose statistical features that can indicate the source words of a blend, and whether an unknown word was formed by blending. We present computational experiments that show the usefulness in these tasks of features tapping into the recognizability of the source words in the blend, in combination with their semantic properties.

1 Lexical Blends

Many natural language processing tasks depend on lexical resources that provide usage information for the words a system must handle. Since words can change in their usage over time, much research in computational linguistics has investigated the automatic learning of lexical information from existing resources, such as corpora, in order to keep lexicons accurate and up-to-date. Most lexical acquisition work has focused on learning syntactic and coarse-grained semantic properties of known words (e.g., Riloff and Jones, 1999; McCarthy, 2001; Korhonen, 2002). However, neologisms—newly created words, which enter the language on a regular basis—present a serious challenge for such efforts, since, by definition, we are not simply adding to or updating our existing knowledge about a word.[1] While available usage data may give strong clues as to the syntactic properties of neologisms, automatically determining their semantics poses a real obstacle to maintaining wide-coverage lexical resources.

Fortunately, people rarely create completely new words that give no clue as to their meaning. One common means for introducing novel words is subtractive word formation processes, in which existing words are clipped and possibly combined. Such methods include shortening (*lab* for *laboratory*), forming an acronym (*CL* for *computational linguistics*), or lexical blending (e.g., Algeo, 1977; Kreidler, 1979). In blending, two or more source words are combined to form a novel word—a blend—which captures some aspect of the semantics of both source words. Blending accounts for roughly 5% of the neologisms in the list of new words accumulated by Algeo (1991). In our study, we focus on two-word sequential blends, such as *brunch* (*breakfast* + *lunch*) and *ebonics* (*ebony* + *phonics*), which are very common type of blend. In these, a prefix of the first source word precedes a suffix of the second.[2]

Many existing English blends, such as *motel* (*motor* + *hotel*) and *meld* (*melt* + *weld*), would be included in a typical computational lexicon. Indeed, some such blends have become conventionalized to the point where some speakers do not even recognize them as blends; however, new blends are continually entering the language, and at an apparently increasing rate (Lehrer, 2003). Blending is used extensively to create novel proper nouns, especially the names of celebrities (*Brangelina*, *Brad Pitt* + *Angelina Jolie*) and companies or organizations (*Fruitopia*, *fruit* + *utopia*, and *Wikipedia*, *wiki* + *encyclopedia*). However, common nouns and other parts of speech are also frequently coined using blending;

[1] Here we do not consider the use of an existing word in a new sense or part of speech as a true neologism, since such usages inherit much from the prior usage(s) of the word.

[2] It is rare for blends to be non-sequential, as in *burble* (*bubble* + *murmur* (Algeo, 1977)), or composed of more than two source words, as in *turducken* (*turkey* + *duck* + *chicken*).

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 289–297,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

recent creations include *mathlete* (*math* + *athlete*), *chillax* (*chill* + *relax*), and *badong* (*bad* + *wrong*).

Automatically learning the meaning of novel lexical blends involves a sequence of challenging steps. First, given an unknown word, it must be decided whether the word is a blend. Second, once the word has been identified as a blend, its source words must be determined. Finally, the sense contributed by each source word, and the semantic relationship between them, must be identified. To our knowledge, none of these three steps has been addressed in the CL literature. In this study, we focus primarily on the second step of this process, that is, the problem of automatically determining the source words of a word that has been identified as a blend. We then present some preliminary results which indicate that our methods for this step can also form the basis for the first step of deciding whether an unknown word is indeed a blend.

2 Computational Models of Blends

Given an unknown word that we know is a blend, the goal is to identify the source words from which it is formed. Currently we assume that we are given no usage information for the word; we consider the possible impact of this assumption when we discuss our experimental results.

Our approach has two steps. First, we identify the set of all pairs of words which, based on orthography, could have formed the blend. We refer to this set as the *candidate set*, and the word pairs it contains as the *candidate pairs*. Then for each candidate pair we calculate a number of statistical features which are expected to be higher for the correct candidate pair and lower for others.

2.1 Finding Candidate Pairs

To create the candidate set for a given blend, we begin by considering each way of splitting the blend into a prefix and suffix.[3] We require that both the prefix and suffix be of length at least two to prevent the candidate sets from becoming excessively large. For example, for the blend *boatel* (*boat* + *hotel*) we consider each of the following prefix–suffix pairs: *bo*, *atel*; *boa*, *tel*; *boat*, *el*.

For each prefix–suffix pair, we extract from a dictionary all words beginning with that prefix and all words ending with that suffix. The Cartesian product of the two sets of words yields a set of candidate pairs. The candidate pairs for all prefix–suffix pairs are combined to form the full candidate set for a blend.

[3] Here we are using the terms *prefix* and *suffix* in the string sense as opposed to the affix morphology sense.

2.2 Statistical Features

Here, we describe our proposed features for selecting the correct candidate pair of source words for a blend. The features are intended to capture the properties of potential source words that make them good components of a blend, and thus are motivated by linguistic properties of blends and cognitive factors in their processing. In this work we are not trying to construct a cognitive model of human interpretation of blends. However, blends are coined by humans operating within certain cognitive conditions (such as constraints on memory and pattern recognition), and these restrictions may give rise to observable statistical properties in the resulting blends. Even though the task of identifying the source words of a blend is sometimes difficult for humans, a computational system should still be able to exploit these properties in automatically performing this task.

2.2.1 Recognizability (REC)

We propose a set of 5 features that relate to the ability of a language user to recognize the source words in a blend. First, it has been noted that frequent items tend to undergo subtractive word formation processes; specifically for blends, this is likely because frequent source words are more easily recognizable (Lehrer, 2003). To reflect this, we include $freq(w_1)$ and $freq(w_2)$, the frequency of each of the two candidate source words, as our first two REC features.

Recognizability of a source word also depends on how much of the word is present in the blend. This is not so much a matter of absolute amount, but rather an effect of neighborhood size. A source word's neighborhood consists of those words that share the portion of the source word that is contributed to the blend (Lehrer, 2003). A source word with a smaller neighborhood is more easily recognized because there are fewer potential words compatible with the blend. Gries (2006) bases his corpus-based study of blends on the related notion of *recognition point*—the minimum amount of material a reader or listener requires to uniquely identify a word. He approximates the recognition point of a word as the minimum length prefix/suffix such that the word is the most frequent in the set of all words which begin/end with

that prefix/suffix. This notion also clearly emphasizes a comparison between the candidate source word and other potential source words.

Following these ideas, we assume that a source word's recognizability is correlated with the degree to which its frequency dominates those of other words with the same prefix/suffix which the source word contributes to the blend We capture this with the following two REC features:

$$\frac{freq(w_1)}{freq(prefix)} \quad (1)$$

$$\frac{freq(w_2)}{freq(suffix)} \quad (2)$$

where $freq(prefix)$ [$freq(suffix)$] is the sum of the frequency of all words beginning with *prefix* [ending with *suffix*].

Recognizability may also be reflected in the degree to which the source words are associated with each other. Blends often describe a conjunction of concepts (e.g., a *spork* is a *spoon* and *fork*), or correspond to a phrasal sequence (e.g., *permalloy* is *permeable alloy*) (Algeo, 1977). We therefore expect that more recognizable blends will be those whose source words are used together relatively frequently. We include as our fifth REC feature the pointwise mutual information (PMI) of the source words, a measure that captures whether two words co-occur more often than expected by chance.[4] Here we assume that two words co-occur if they appear in the same sentence.

2.2.2 Contribution and Length (CONT/LEN)

We use 4 features which capture properties relating to the orthographic length of a blend's source words and the amount of material they contribute to the blend.

Our first CONT/LEN feature comes from Gries's (2004) finding that the second source word tends to be longer than the first, in terms of either graphemes or phonemes. We capture this as a feature according to the following formula where $len(w)$ is the length of w in graphemes.

$$\frac{len(w_2)}{len(w_1)+len(w_2)} \quad (3)$$

Gries also notes that the second source word will tend to contribute more of itself to the blend than the first source word. Considering the contribution in terms of graphemes, we first define the *contribution* of a word w to a blend b as:

$$cont(w,b) = \frac{len(prefix\ or\ suffix\ of\ w\ in\ b)}{len(w)} \quad (4)$$

We then define the second CONT/LEN feature as:

$$\frac{cont(w_2,b)}{cont(w_1,b)+cont(w_2,b)} \quad (5)$$

Gries further finds that the shorter source word will tend to contribute more than the longer source word. We encode this as the third CONT/LEN feature according to the formula below, which is positive when the shorter source word contributes more and negative otherwise.

$$[len(w_1)-len(w_2)]*[cont(w_2,b)-cont(w_1,b)] \quad (6)$$

Kubozono (1990) finds evidence that the length of a blend will tend to be similar to the length of its second source word. Our approximation yields the fourth CONT/LEN feature:

$$1-\frac{|len(w_2)-len(blend)|}{max(len(w_2),len(blend))} \quad (7)$$

2.2.3 Phonetics (PHON)

We include 3 features motivated by phonetic similarity. It is expected that source words are phonetically similar to the resulting blend (Gries, 2006). Since we would like our features to be calculable for truly novel blends for which we may not have a phonetic representation, here we approximate phonetic similarity by orthographic similarity. We calculate the longest common subsequence of each of the candidate source words and the blend for our first two PHON features.

The source words of a blend also often have a noticeable degree of phonetic overlap with each other (Algeo, 1977; Gries, 2006). We use the longest common subsequence of the two candidate source words as our third PHON feature. Here, as in Gries (2006), we use the phonetic transcription of the source words, since it is assumed that the source words (unlike the blend) are known words that do occur in a lexical resource.

2.2.4 Part of Speech (POS)

Since many blends are similar to conjunctions, we expect that the source words which form a blend will often be of the same part of speech (POS). Our single POS feature is an estimate of

[4] $PMI(x,y) = \frac{log p(x,y)}{p(x)p(y)}$

the probability of each candidate source word pair, w_1, w_2, occurring as the same coarse-grained POS according to the formula below.

$$\sum_{pos_i \in \{N,V,A,O\}} \frac{freq(w_1, pos_i)}{\sum pos_{j \in \{N,V,A,O\}} freq(w_1, pos_j)} * \frac{freq(w_2, pos_i)}{\sum pos_{j \in \{N,V,A,O\}} freq(w_2, pos_j)} \quad (8)$$

where $freq(w, pos)$ is the frequency of word w occurring as POS pos, and N, V, A, and O are noun, verb, adjective, and other, respectively.

2.2.5 Semantics (SEM)

Lehrer (2003) notes that people can more easily identify the source words of a blend when there is a semantic relation between them. We formulate two SEM features to incorporate this type of knowledge into our identification method.

As noted, blends are often composed of two semantically similar words, reflecting a conjunction of their concepts. Thus, a *spoon* and *fork* are both eating utensils that combine to form a *spork*, while *melt* and *weld* both involve the application of heat, combining to form *meld*. Our first SEM feature captures semantic similarity using an ontological similarity measure, which is calculated over an ontology populated with word frequencies from a corpus.

The other common type of blend corresponds to the blending of sequential phrases, as noted earlier in *permalloy*. Here, the source words are not necessarily similar, but are typically semantically related since they can form a felicitous phrase. Our second SEM feature is a measure of semantic relatedness using distributional similarity between word co-occurrence vectors.

2.3 Scoring the Candidate Pairs

To determine the source words for a blend, we sum the features described above, or some subset of them, to yield a score for each candidate pair.[5] In our experiments, we calculate the score separately using each of the 5 feature groups—REC, CONT/LEN, PHON, POS, SEM—as well as some combinations of them. Candidate pairs are then ranked according to the score given by the features under consideration, and some number of the highest ranked pairs are returned by the system.

[5]To ensure that the features have the same range of values and can be meaningfully summed, we normalize the features in each candidate set and take an arctan transform.

Note that a candidate set may contain the same candidate pair multiple times, corresponding to different ways of splitting the blend, as in *boatel* formed from *boa(t)+(ho)tel* or *boat()+(hot)el*. Although the source words are the same, their split into a prefix and suffix differs, affecting some of our feature calculations. For such candidate pairs, we consider only the maximum score attained across its possible prefix–suffix splits.

3 Materials and Methods

3.1 Experimental Items

For this initial study, we wanted to use true blends that have been accepted by speakers, with source words objectively determined by lexicographers. Our experimental items thus are existing words from a standard lexical resource, the Macquarie dictionary (MD, Delbridge, 1981).

Using a simple regular expression, we automatically extracted all entries from MD whose etymology included the word *blend*, or indicated that the word was formed by combining two words, at least one of which was clipped. (The latter condition excludes compounds such as *broadband*.) The result of this process was a list of potential blends which we manually filtered to ensure that the items used in this study were truly blends. We removed any items for which either of the source words was an affix (i.e., *un-*, *-tion*) or did not have an entry in MD. We also eliminated any item whose lexical entry indicated doubt that it is a blend (e.g., a question mark at the beginning of the etymology field). To simplify the problem in this initial study, we imposed the additional constraint that the source words must not be words-with-spaces (e.g., *abietic acid*), and that no additional graphemes may occur between the source words in the blend (e.g., *donkophant*, *donkey* + *elephant*). From the resulting list, we manually extracted a total of 192 two-word sequential blends along with their corresponding source words.

The 192 blends were divided into 3 frequency ranges according to their frequency in the British National Corpus (BNC, Burnard, 2000). Development (DEV) and test (TEST) sets were selected to have an equal proportion of blends from these 3 groups, and contain 95 and 97 items respectively.[6]

[6]In the experiments reported here, we do not consider the effect of the differing frequency ranges.

3.2 Candidate Sets and Features

To create the candidate sets we used all words in MD as our list of potential source words. We used frequency information from the BNC to calculate the word and prefix/suffix frequencies for the REC features. The CONT/LEN features were calculated from the spelling of the words. Phonetic transcriptions of each candidate source word were taken from MD, which we use (without stress markers) to calculate the PHON features. We use the BNC to determine the frequency of a word occurring as one of the 4 coarse-grained parts of speech for the POS features. We compute the first SEM feature, semantic similarity, using Jiang and Conrath's (1997) measure from the WordNet::Similarity package (Pedersen et al., 2004). The second SEM feature, semantic relatedness, is calculated as the cosine between word co-occurrence vectors, generated using software provided by Mohammad and Hirst (2006).

3.3 Evaluation Metrics

We evaluate the accuracy of our system as the percentage of items for which the correct candidate source word pair—as determined by the gold standard (MD)—is scored highest. However, this is a very stringent evaluation measure given the size of the candidate sets (ranging from 0 to over 2M, with averages on the datasets of 17K and 29K). To be of use in a semi-automated system, having the correct source word pair reasonably high in the scoring would be sufficient. We thus also evaluate our system according to an *in-top-5* accuracy metric, where the output of the system is deemed correct if the correct source word pair is in the top 5 pairs returned by the system. We compare these two system accuracies, *in-top-1* and *in-top-5*, to a uniform random baseline of $\frac{1}{size\ of\ candidate\ set}$ and $\frac{5}{size\ of\ candidate\ set}$, respectively.[7]

Given the importance of word frequency in blend formation, we also compare our system against an informed baseline which uses only two features—the frequency of each candidate source word. (Note that this baseline includes 2 of the 5 REC features.)

We do not have human judgments of source word identification on our blends, and so cannot provide an expected upper bound on performance for this particular set of blends. However, psycholinguistic research indicates that source word identification is a difficult task. Lehrer (2003) shows that human subjects achieve between 34% and 79% accuracy in source word identification, depending on the amount of material the source words contribute to the blend. When calculating relative error reduction, we consider the upper bound to be 100%, since the variation of human performance on this task does not indicate a clear upper bound.

4 Results on Source Word Identification

In Section 4.1 we report results on both DEV and TEST for identifying the source words from which a blend is formed. Because of the frequency of "conjunctive" blends (see Section 2.2), especially noun-noun blends like *brunch*, in Section 4.2 we also look at the performance of our methods on the subset of our blends formed from two nouns. We also have concerns that some of our experimental items are not actually blends.[8] In Section 4.3 we consider results on the subset of items which were confirmed to be blends in other lexical resources.

4.1 Results on All Blends

The first panel of Table 1 shows the results on DEV and TEST for both baselines and for the feature groups which outperform the frequency baseline, REC and SEM. The accuracies obtained using the CONT/LEN, PHON, and POS features are mostly better than the random baseline, but do not outperform the frequency baseline.

The first panel of Table 1 also shows the results on two combinations of features. We combine the REC features with the SEM features, since they perform best alone. This approach is also supported by the observation that individual source word recognizability and semantic compatibility work together in the human identification of blend source words (Lehrer, 2003). The results using REC+SEM are better than the frequency baseline, but are not substantially better than the REC features. We also combine all the features, shown as ALL. This gives results similar to those obtained for REC and REC+SEM.

In the rest of this section we focus on the REC, REC+SEM, and ALL features, since these give the best results. Indeed, given the low random baseline of 1% on the *in-top-1* task and a likely upper

[7] We eliminate duplicate word pairs from the candidate sets before calculating their size.

[8] For example, it does not match our intuition that *clash* is a blend of *clap* and *dash*.

bound on human performance of 34–79%, the accuracies of 12–26% using these feature groups are a promising start.

4.2 Results on Noun–noun Blends

Since some of our features, such as semantic similarity, are based on the notion that a blend's source words are likely to be of the same POS, we expected our methods would perform better on noun–noun blends. We consider a word to be a noun if its predominant POS in the BNC was noun. $\text{DEV}_{\text{N–N}}$ and $\text{TEST}_{\text{N–N}}$ include all of the blends formed from two nouns from DEV and TEST, respectively. Each new dataset contains 53 items.

Results on noun–noun blends are shown in the second panel of Table 1. The performance of the feature groups shown is better than the frequency baseline, and also always better than that of the corresponding experiment on DEV or TEST (first panel of Table 1). Note that the frequency baseline for noun–noun blends is generally higher than that for all blends. Nevertheless, when we consider relative error reduction over the frequency baseline, the performance of REC+SEM is always higher on noun–noun blends than on all blends. This indicates that this combination of features does in fact perform better on noun–noun blends.

4.3 Results on Confirmed Blends

We extracted all items from DEV and TEST that are noted as a blend in at least one of two other lexical resources (Soanes and Stevenson, 2004; Mish, 2003). This process resulted in 29 and 30 items in DEV_{Conf} and $\text{TEST}_{\text{Conf}}$, respectively.

Results on these blends are shown in the third panel of Table 1. When comparing these results to those on all blends (first panel of Table 1), we see that for the random baseline, frequency baseline, and REC features, the results are quite close for the corresponding dataset (DEV or TEST) across the two panels. The very consistent values for these basic results across the full and reduced datasets indicate that although the Conf datasets are small they are reasonably representative.

The results using the combination of all features are not consistently higher across the DEV_{Conf} and $\text{TEST}_{\text{Conf}}$ datasets. However, the results using the combination of REC+SEM stand out in that they show a substantial increase in performance for both datasets over results using just the REC features. There are two possible explanations for why this is the case. One possibility is that the items which were not confirmed to be blends in another resource are in fact not blends (i.e., the etymology entries in MD are incorrect). In this case, the better performance of the REC+SEM features would confirm that (true) blends are indeed formed from words which tend to be semantically similar. The second possibility is that the unconfirmed items actually are blends (i.e., the other lexical resources either did not include these items or did not correctly identify their etymology). In this case, the higher performance of the REC+SEM features would suggest that lexicographers are more likely to recognize that a word is a blend when its source words are semantically similar.

4.4 Discussion

We have shown that the REC features, especially in combination with the SEM features, achieve good results on all the blends, and even better results on subsets of blends with particular properties. The complementarity of REC and SEM is in line with observations of the importance of a semantic connection between source words when one of the words is difficult to recognize (Lehrer, 2003). Although our *in-top-1* numbers are low in absolute terms, they show reductions in error rate of up to 20%, on a task that is known to be very difficult for humans (with performance varying between 34% and 79% depending on the dataset). Our *in-top-5* performance, reaching near or over 50% in a number of cases, indicates promise for a semi-automated assistant for lexicographers.

It is perhaps not surprising that our rough phonetic approximations of PHON do not perform well in identifying source words; similarly, our POS features are grounded on the fact that many blends use source words that have the same POS, but many do not. It is more surprising that the CONT/LEN features do not perform well. Corpus studies (Gries, 2004, 2006; Kubozono, 1990) have found consistent patterns of source word contribution and length which our features are drawing on. Interestingly, Lehrer (2003) observes that the amount of contribution of a source word to a blend is only a factor in human identification of source words for *unknown* blends, not blends that are familiar to the subjects. It is not clear though why this would affect the resulting pattern of contribution across blends, which our score is based on.

Our performance may be limited because we

Table 1: % in-top-1 and in-top-5 accuracy on blends in DEV and TEST.

	All Blends				Noun–Noun Blends				Confirmed Blends			
	DEV		TEST		$DEV_{N–N}$		$TEST_{N–N}$		DEV_{Conf}		$TEST_{Conf}$	
Features	top1	top5	top1	top5	top1	top5	top1	top5	top1	top5	top1	top5
Rand baseline	1	6	1	6	2	7	1	7	2	8	1	5
Freq baseline	6	27	14	31	8	36	15	30	10	28	13	33
REC	12	34	18	43	15	43	19	49	17	38	17	43
SEM	12	25	15	33	-	-	-	-	-	-	-	-
REC+SEM	17	31	26	43	23	40	32	55	28	45	27	47
ALL	17	34	15	39	25	40	25	45	28	52	13	43

consider the blends without any contextual information. Since people can identify the source words of blends much more easily in context (Lehrer, 2003), we are likely missing important clues to the source words by taking this approach. A next step is to use the distributional similarity between the context of a blend and its source words (and their contexts) to address this issue.

5 Results for Blend Identification

Our focus here has been on identifying the source words of a word known to be a blend. However, given the connection between our features and the factors that influence the creation of blends (Section 2.2), these same features may be useful in determining whether a word is in fact a blend. Our features were designed to have higher values for the candidate pair which formed a blend, and lower values for others. We therefore expect that the top score for a candidate source word pair for a blend will be higher than the top score for a non-blend which has no correct source word pair.

To test this hypothesis we perform experiments in which we compare our candidate pair scores across both blends and non-blends. To calculate the scores here, we use the REC features only, since adding the semantic information did not help on all our blends. For the known blends we use our full DEV and TEST sets. For the non-blends, we created two datasets of non-blends, DEV_{NB} and $TEST_{NB}$. These datasets were formed by randomly sampling words from MD which were not blends according to our heuristics. The words were chosen to yield the same proportion of items in the three frequency ranges (Section 3.1), and the same proportion of words from three length ranges, as those in DEV. The resulting DEV_{NB} and $TEST_{NB}$ sets contain 98 and 99 words, respectively.

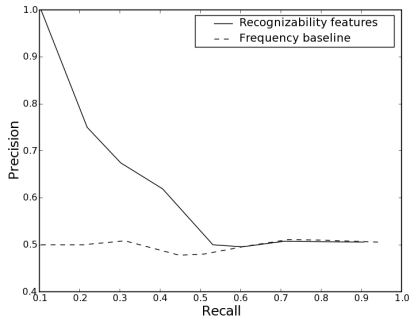


Figure 1: Interpolated precision-recall curve for REC features and frequency baseline on items in TEST and $TEST_{NB}$.

We created candidate sets and calculated the features for the items in DEV_{NB} and $TEST_{NB}$ the same way we did for DEV and TEST. For each item, we then took the maximum of the score assigned to any of its candidate pairs.

The goal is to use our statistical score to separate blends from non-blends. We merge all items and determine how accurately the scores pick out the blends. That is, we rank all the blends and non-blends (together) by score, then consider higher scores to indicate that an item is a blend. We measure the precision for a fixed value of recall, and compare the results using the REC features against the frequency baseline features. The precision–recall curves for items in TEST and $TEST_{NB}$ are shown in Figure 1; results on DEV and DEV_{NB} are similar. This figure shows that our method for determining whether an unknown word is a blend does tend to assign higher scores to blends than non-blends, and substantially outperforms the frequency baseline up through a recall value of 0.4.

6 Related Work

Much research in CL on unknown words has focused on determining various syntactic properties (e.g., Mikheev, 1997; Peng et al., 2004). Some work has tried to give a coarse semantic characterization of words unknown to the application (Toole, 2000). Our perspective here is quite different: we identify a highly productive word creation process (that of blending), and use the linguistic properties of blends to develop automatic means for their identification and interpretation. To our knowledge, we are the first to do so.

A related subtractive word formation process that has been investigated recently in the CL community is that of acronyming. Automatically inferring the expanded form of acronyms has received particular attention in the bio-medical domain. This problem is similar to that of identifying the source words of a blend in that the words in the expanded form must be identified from their orthographic contribution to the acronym. In fact, approaches to determining expanded forms, such as Schwartz and Hearst (2003) and Okazaki and Ananiadou (2006), are similar to ours in that they construct a set of candidate expanded forms and then attempt to select the most appropriate one from this set. However, since each word in an expanded form typically contributes only its first letter to the acronym, there is less information available to determine the expanded form than there is to infer the source words of a blend. On the other hand, the canonical form of acronym definitions and the domain specificity of the problem—two properties which have no counterpart in our work on lexical blends—can be exploited.

7 Conclusions

We identify a highly productive and very common novel word creation process—lexical blending—and investigate techniques for determining the source words of a blend based on the linguistic and cognitive properties of this process. Given a novel blend like *chillax*, the goal is to discover that this word is formed from *chill+relax*. This is a crucial step in determining the semantics of a large (and growing) class of neologisms. Our results indicate that statistical features capturing the recognizability of the source words consistently perform well in their identification. For blends with certain properties, performance is improved by the addition of features tapping into the semantic similarity and relatedness of the source words. Our results range from 12% to 32% on a task with an informed baseline of 6–15% and which human subjects find difficult. We also show that our features may be useful in determining that an unknown word is a blend.

Still, there is much room for improvement, as well as extensions to the work. First, we plan to explore the use of contextual information, which is critical to human performance on the source word identification task (Lehrer, 2003). Next, we must build on our work to tackle the next step of determining which sense each source word contributes to the blend, and how they are related, in order to achieve our goal of learning the semantics of a blend.

References

John Algeo. 1977. Blends, a structural and systemic view. *American Speech*, 52(1/2):47–64.

John Algeo, editor. 1991. *Fifty Years Among the New Words*. Cambridge University Press, Cambridge.

Lou Burnard. 2000. *The British National Corpus Users Reference Guide*. Oxford University Computing Services.

Arthur Delbridge, editor. 1981. *The Macquarie Dictionary*. Macquarie Library.

Stefan T. Gries. 2004. Shouldn't it be breakfunch? A quantitative analysis of the structure of blends. *Linguistics*, 42(3):639–67.

Stefan T. Gries. 2006. Cognitive determinants of subtractive word-formation processes: A corpus-based perspective. *Cognitive Linguistics*, 17(4):535–58.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference Research on Computational Linguistics (ROCLING X)*.

Anna Korhonen. 2002. Semantically motivated subcategorization acquisition. In *Proceedings of the ACL Workshop on Unsupervised Lexical Acquisition*, pages 51–58.

Charles W. Kreidler. 1979. Creating new words by shortening. *English Linguistics*, 13:24–36.

Haruo Kubozono. 1990. Phonological constraints on blending in English as a case for phonology-

morphology interface. *Yearbook of Morphology*, 3:1–20.

Adrienne Lehrer. 2003. Understanding trendy neologisms. *Italian Journal of Linguistics*, 15(2):369–382.

Diana McCarthy. 2001. *Lexical Acquisition at the Syntax-Semantics Interface: Diathesis Alternations*. Ph.D. thesis, University of Sussex.

Andrei Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Computational Linguistics*, 23(3):405–423.

Frederick C. Mish, editor. 2003. *Merriam-Webster's Collegiate Dictionary*. Merriam-Webster, Incorporated, eleventh edition.

Saif Mohammad and Graeme Hirst. 2006. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)*, pages 35–43.

Naoaki Okazaki and Sophia Ananiadou. 2006. A term recognition approach to acronym recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (Coling-ACL 2006)*, pages 643–650.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *HLT-NAACL 2004: Demonstration Papers*, pages 38–41.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 562–568.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 474–479.

Ariel S. Schwartz and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical texts. In *Proceedings of the Pacific Symposium on Biocomputing (PSB 2003)*, pages 451–462.

Catherine Soanes and Angus Stevenson, editors. 2004. *Concise Oxford English Dictionary*. Oxford University Press, eleventh edition.

Janine Toole. 2000. Categorizing unknown words: Using decision trees to identify names and misspellings. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 173–179.

Discovering Word Senses for Polysemous Words Using Feature Domain Similarity

Noriko Tomuro
DePaul University, School of Computer Science, Telecommunications and Information Systems,
Chicago, IL 60604 USA
tomuro@cs.depaul.edu

Kyoko Kanzaki, Hitoshi Isahara
National Institute of Information and Communications Technology,
Soraku-gun, Kyoto, 619-0289 Japan
{kanzaki,isahara}@nict.go.jp

Abstract

This paper presents a new clustering algorithm called *DSCBC* which is designed to automatically discover word senses for polysemous words. DSCBC is an extension of CBC (Pantel and Lin, 2002), and incorporates *feature domain similarity*: the similarity between the features *themselves*, obtained a priori from sources external to the dataset. By incorporating the feature domain similarity in clustering, DSCBC produces *monosemous clusters* (a cluster in one domain), thereby discovering individual senses of polysemous words. For evaluation, we apply the algorithm to Japanese and English *adjectives*, and compare the derived senses against manually created lexicons. The results show significant improvements over other clustering algorithms including CBC.

1 Introduction

Automatic acquisition of word senses from corpora has been an active topic of research in Natural Language Processing (NLP). Most conventional thesauri are manually compiled by lexicographers, therefore often contain inconsistencies. Word senses are also sometimes too fine-grained or specific, and not based on the common usage of the language, as frequently pointed out for WordNet (Miller, 1990). A better approach is to automatically derive word meanings from real data.

In the NLP research, many techniques have been developed which discover word senses or semantic classes from corpora (e.g. (Hindle, 1990; Lin and Pantel, 2001)). However, most of them have focused on nouns, which are largely monosemous, so the applicability of those techniques to polysemous words such as verbs and adjectives is unknown. Walde (2006) and Boleda et al. (2004) present clustering experiments which automatically derive semantic classes for German verbs and Catalan adjectives respectively. However, they both use standard clustering methods (such as K-means) only, and no attempt is made to develop new techniques tailored to polysemous words.

Pantel and Lin (2002) develop a clustering algorithm called *CBC* (Clustering By Committee), and report a superior performance of CBC over other clustering algorithms in discovering word senses from texts. However, their results are shown again mostly for nouns, and in fact, the algorithm does not include any special considerations to avoid generating *polysemous clusters*: a cluster in which several word meanings are mixed in. The problem of polysemous clusters also exists in other clustering algorithms when data contains many polysemous words. Indeed, some of the semantic classes derived in (Walde, 2006) are polysemous clusters.

In this paper, we present a new clustering algorithm called *DSCBC*, which is specifically designed to automatically discover senses of polysemous words (but works for monosemous words as well). DSCBC is an extension of CBC, and incorporates *feature domain similarity*: a prior knowledge about the similarity between features *themselves*. For example, if data instances x and y are represented by features $f1$ and $f2$, the feature domain similarity refers to the similarity between $f1$ and $f2$ (not between x and y). Feature domain similarity is obtained a priori from sources external to the dataset(s) used at hand. By incorporating the feature domain similarity in clustering, we are able to group data instances which have features in the same or similar *domain*, thereby producing monosemous clusters.

To test the effectiveness of DSCBC, we apply the algorithm to Japanese and English adjectives. In this work, we use *abstract nouns* as the features to represent adjectives. From a large cor-

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 298–307,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

pus, we collect many instances where adjectives modify abstract nouns (e.g. "happy feeling") and construct a dataset for each language. Then after applying the algorithm, we compare the automatically derived word senses against manually created lexicons (*Daijirin* for Japanese and WordNet for English). The results show significant improvements by DSCBC over other clustering algorithms in discovering polysemous adjective senses.

2 Semantic Categories of Adjectives

In linguistics, there is a large body of work on adjectives. However, the attention they have received is much less than that for nouns and verbs. Likewise in NLP, there exist little other works which tackled adjectives specifically. Adjectives are usually considered in the context of nouns which they modify, and thought to add only auxiliary information to the nouns. Moreover, adjectives are generally difficult to incorporate in NLP applications because of their polysemous nature: not only do adjectives change or shift meanings depending on the nouns they modify (e.g. "warm temperature" vs. "warm person"), they are often used metaphorically (e.g. "dark conversation" – a discussion on grim topics, or to talk pessimistically). Most of the recent work on adjectives in NLP has focused either on specific applications (e.g. classifying documents according to subjectivity or sentiment (Wiebe, 2000)), or on specific types of adjectives (e.g. event adjectives (Lapata, 2001) and gradable adjectives (Hatzivassiloglou and Wiebe, 2000)).

Traditionally, meanings of a polysemous word are enumerated in dictionaries. For example, WordNet lists ten senses for "warm". Distinguishing different senses of polysemous words is a difficult problem for any part of speech. However for adjectives, salient homonym-level meanings are distinguishable based on the *domains* of the co-occurring nouns (Justeson and Katz, 1995).

Note that the premise of sense enumerability can be argued. Notably, Pustejovsky (1995) proposes a generative view of adjectival meanings: meanings of adjectives are fundamentally indifferentiable, and a specific meaning is generated (by coercion) when an adjective is combined with a noun in the context. However, knowing the specific meaning of an adjective in the given context would greatly help the understanding of the sentence in NLP tasks and applications, such as information extraction and question-answering. Our work in this paper aims to acquire such specific adjectival meanings that arise in the context as abstract *semantic categories* or *classes* of adjectives, automatically from corpora. Then the derived semantic adjectival categories can be used in constructing lexicons as well as investigating the polysemy of adjectives linguistically.

3 Datasets

In this work, we use *abstract nouns* as the features to represent adjectives, such as "gentle personality" (as versus concrete nouns such as "gentle goats"). Previous studies in linguistics have shown that, when an abstract noun is modified by an adjective, it often correlates with the semantic class of the adjective (e.g. "gentle" as a personality adjective) (Takahashi, 1975). Therefore, abstract nouns provide the meanings of adjectives literally and immediately, while with concrete nouns, the meanings must be inferred.

In the current work, we applied our sense discovery algorithm to two datasets. The first dataset is Japanese, extracted from a large volume of Japanese newspaper articles, and containing a large number of examples of adjectives with co-occurring abstract nouns. This dataset was previously used in (Kanzaki et al., 2004). From the original dataset, we randomly selected 1500 instances/adjectives to create the dataset. The number of co-occurring nouns/features was 361.

The second dataset is English, extracted from the Web n-gram corpus created by Google Inc. (Brants and Franz, 2006). This corpus contains English word n-grams (uni-grams to five-grams) and their frequency counts obtained from publicly accessible Web pages containing approximately 1 trillion word tokens. To extract adjectives co-occurring with abstract nouns, we first translated the nouns used in the Japanese dataset into English, and obtained a set of 277 English abstract nouns.[1] Then we extracted instances of the forms 'Adj AbN' and 'AbN BE Adj' (where BE stands for a copula verb, for instance "nature is gentle") from the bi-grams and tri-grams respectively.[2] Note that, although neither pattern guaran-

[1] Since there were several cases where two or more Japanese nouns corresponded to the same English noun (and less number of cases the other way around), the number of the English nouns was less than that of the Japanese nouns.

[2] The part of speech for Adj was checked against WordNet – we extracted instances where the word at the respective Adj

tees the matched AbN is truly the noun which the Adj modifies, no outstanding spurious instances were observed. There were over 15000 adjectives which co-occurred with the selected abstract nouns in the corpus. Then we randomly selected 1500 of them to create the dataset.

Besides language, the two datasets have a marked difference: the Japanese dataset is relatively sparse, while the English dataset is very dense. This is probably because newspapers use adjectives less than other genres, while web pages include a variety of genres and authorship.

For both datasets, we represented each adjective by a feature vector, where a feature was an abstract noun co-occurred with the adjective. The value was the *Mutual Information (MI)* computed from the frequency counts in the respective corpus.[3] The MI between two words x and y, denoted $I(x,y)$, is defined as:

$$I(x,y) = log\frac{p(x,y)}{p(x)p(y)}$$

where $p(z)$ is the probability of a word z, and $p(x,y)$ is the joint probability of x co-occurred with y. MI indicates the mutual dependence between two random variables, where $I(x,y) = 0$ if x and y are independent, or a positive value otherwise. In our case, $I(x,y)$ essentially indicates how well a feature noun predicts (or is correlated with) a given adjective. MI has been often used in NLP as a way to put weights on feature values (e.g. (Hindle, 1990)).

4 Clustering By Committee (CBC) Clustering

Our sense discovery algorithm is an extension of *Clustering By Committee (CBC) Clustering* (Pantel and Lin, 2002). We introduce CBC and discuss its limitations in this section, then describe the modifications we made to CBC to derive our algorithm in the next section.

4.1 The CBC Algorithm

CBC is an unsupervised clustering algorithm which automatically derives a set of *committees*. A committee is a cluster of words which are very similar to each other – similar in notion to a synonym set (synset). As with synset, each committee corresponds to a word sense. In CBC, a committee is represented by the centroid vector of the word vectors which comprise the cluster. The algorithm automatically derives committees by first finding a *tight* cluster of words which are similar to a given word, for every word in the dataset, then selecting a subset of the derived clusters whose centroid vectors are dissimilar/orthogonal to each other as committees. After obtaining the committees, CBC discovers the senses of a word by assigning the word to its most similar committees – all committees with which the similarity is above a threshold, and removing the features of the selected committees from the word as they are selected.

Phase I: Find clusters.
Let L be a list of clusters, initially empty.
For each word $w \in W$ in the dataset,
Select a set c of at most n words from W which are the most similar to w, and add c to L if it is not already in L.
Sort L in the descending order of the scores of the clusters.(1)

Phase II: Find committees.
Let C be a list of committees, initially empty.
For each cluster $c \in L$ (in the sorted order),
Compute the centroid of c.(2)
If it is not similar to any other committee in C, add c to C.

Phase III: Assign words to committees.
For each word $w \in W$,
Select all committees in C whose centroids are similar to w.

Figure 1: The CBC Algorithm

Figure 1 shows the overall steps of the CBC algorithm. A cluster found in Phase I is tight in that it consists of at most n similar words (where $n = 10$ is used in (Pantel and Lin, 2002)). Then, clusters are selected into the set of committees in the descending order of their scores (indicated with (1) in the algorithm). This way, CBC guarantees that only the tight clusters which are also well scattered in the feature space are selected as committees. The score of a cluster c is computed as $|c| \times$ avgsim(c), where $|c|$ is the number of words in c, and avgsim(c) is the average pairwise similarity between the words in c. Note that throughout

position is recorded as an adjective in WordNet.

[3] Actually for the English data, we took the log of base 10 of the frequency counts, since the values were significantly higher and their range was much wider than the Japanese dataset.

the CBC algorithm, the cosine coefficient (Salton and McGill, 1983) is used to measure the similarity between vectors, and MI is used to compute the values in the word vectors.

4.2 Limitations of CBC

Pantel and Lin (2002) applied this CBC algorithm to a large corpus (a parsed TREC collection), and reported its superior performance over other standard clustering algorithms as well as their predecessor algorithm called UNICON (Lin and Pantel, 2001). However, one problem with CBC is that it does not work well for polysemous words despite its claimed utility. The reason is because the algorithm uses the centroid to represent a committee (indicated with (2) in the algorithm). Consider the adjectives "warm" and "cold". These words have similar sense patterns, in particular, they both have the senses of temperature and personality. Many other adjectives which elaborate temperature such as "cool" have those two meanings as well – the pattern of temperature-personality is rather regular in adjectives (i.e., *systematic polysemy* (Pustejovsky, 1995)). Since their word vectors are similar, they are most likely grouped in the same cluster. And if this cluster is selected as a committee (where the algorithm has no mechanisms besides cosine to prevent it), the committee centroid will end up having both senses, thus failing to separate the two individual senses.

For CBC to produce monosemous committees, either the data contains mostly monosemous words, or sense patterns of the polysemous words in the data are not regular so that they won't constitute tight clusters. In the case of adjectives, words are highly polysemous,[4] and there are many patterns of polysemous senses that are regular.

The problem of polysemous clusters also arises in other clustering algorithms when the data contained many polysemous words. The problem will happen in other methods besides clustering in the same way as well. For example, Principal Component Analysis (PCA) finds a set of principal components (linear combinations of features) which account for the variance in the data. Principal components are orthogonal to each other, and in that sense, CBC has the same objective as PCA. However again, features involved in the regular polysemy patterns of the polysemous words are grouped into the same principal component, resulting in a heterogeneous, polysemous component in PCA.

[4]In WordNet, the proportion of polysemous words is 31% for adjectives, as compared to 16% for nouns. In our English dataset, 723 (= 48%) of the 1500 adjectives had more than one sense in WordNet.

5 Domain Similarity CBC (DSCBC) Clustering for Polysemous Words

5.1 Feature Domain Similarity

To overcome the polysemous cluster problem discussed above, we developed a new metric which measures the similarity between the features themselves, which we call the *Feature Domain Similarity*. This is the primary characteristics and the contribution of our algorithm.

Consider the following data:

```
   a b c d
x: 1 1 0 0
y: 1 0 1 0
z: 1 0 0 1
```

where x, y, z are data instances, and a, b, c, d are features. In most clustering algorithms, features are assumed to be independent to each other, or their dependencies are ignored. So in the example, x is equally likely clustered with y or z, because the similarity between x and y, and x and z are the same (based on the Euclidean distance, for example). However if we have a priori, general knowledge about the features that b's domain is more similar to that of c than to d, it is better to cluster x and y instead of x and z, because the $\{x,y\}$ cluster is "tighter" than the $\{x,z\}$ cluster with respect to the domains of the features.

Feature domain similarity can be obtained from any linguistic resources (but should be external to the dataset(s) to be used in the experiments). For Japanese, we used the case frame data extracted automatically from the Web (Kawahara and Kurohashi, 2006). In our work, features are abstract nouns. So for each abstract noun (e.g. *kimochi* "feeling"), words which appeared in various case frame relations with the noun (largely verbs and adjectives, e.g. *kimochi ni naru* "come to bear feeling", *kimochi wo tsutaeru* "convey feeling") were used to represent the noun.[5] Then the cosine coefficient was used to compute the similarity between the nouns. For English, we used *javasimlib*:[6] a Java-based tool which computes the similarity between words (or synsets) over the Word-

[5]http://reed.kuee.kyoto-u.ac.jp/cf-search/

[6]http://wordnet.princeton.edu/links#extensions

Net hierarchies based on an information theoretic metric (Seco et al., 2004). Given two words or synsets, *javasimlib* returns a value between 0 and 1, where 1 indicates the highest similarity. For each noun, we considered the top (at most) two senses in the calculation in order to avoid spurious results arising from rare or unimportant senses encoded in WordNet. Then we defined the similarity between two nouns as the maximum value between their top two senses, that is,

$$fdsim(a,b) = \arg\max_{i,j\in\{1,2\}} jsim(a\#i, b\#j)$$

where a, b are nouns, $fdsim(a,b)$ denotes the feature domain similarity between a and b, $a\#i$ is the ith sense of a, and $jsim(x,y)$ is the similarity between two synsets x and y returned by *javasimlib*. Then for each dataset, we computed all pairwise similarities between the nouns/features and stored them in a matrix.

Next, we defined a notion of *domain tightness* for a single word vector. This metric indicates how "tight" the vector is with respect to the domains of its features (i.e., the degree of monosemy). For a given word vector v, the domain tightness of v, denoted $dt(v)$, is define as:

$$dt(v) = \frac{1}{n} \sum_{a,b\in F; a\neq b} \frac{v(a)+v(b)}{2} fdsim(a,b)$$

where F is the set of features used to represent v, $v(x)$ is the value in v for the feature $x \in F$, $fdsim(a,b)$ is the feature domain similarity between a and b described above, and n is the number of pairwise combinations of the features in F where their feature-values are greater than zero (i.e., n = count($\langle$a,b$\rangle$: $v(a) > 0$ and $v(b) > 0$)). Thus it is the average feature domain similarity between two non-zero features in v, weighted by the average of the two feature-values.

Finally, we defined a new similarity metric between two word vectors that incorporates the feature domain similarity, which we call *dsSim*, as follows. For given word vector $v1$ and $v2$, the similarity between them, denoted $dsSim(v1,v2)$ is:

$$dsSim(v1,v2) = w_0 \times cos(v1,v2) + w_1 \times dt(v3)$$

where $cos(v1,v2)$ is the cosine between $v1$ and $v2$, $v3$ is a vector in which $v1$ and $v2$ are merged (i.e., $v3 = v1 + v2$), and w_0, w_1 are weights which sum up to 1. We used $w_0 = 0.95$ and $w_1 = 0.05$ in our experiments. This new measure is essentially based on cosine, added with the domain tightness of the merged vector. We looked at the merged vector, because a cluster centroid is a vector in which all member word vectors are merged (and averaged), and that's precisely what we wanted to improve upon CBC – the domain tightness of the cluster/committee centroids.

5.2 The DSCBC Algorithm

In addition to feature domain similarity, we also incorporated a few more ideas which may help derive monosemous committees. One idea is to limit the number of features to be used in the committee centroids. By taking the top k features, only the most salient features are kept (which hopefully are features in one domain), and other insignificant features are discarded. Another idea is to use only the features that appeared with the majority of the words which make up the committee.

Furthermore, we derived the committees incrementally in two steps: first produce committees from the original dataset, then derive the final set of committees from those produced in the first step. This hierarchical scheme is aimed to find the second or higher-order features (such as those in a transitive similarity relation).

We present our algorithm named DSCBC (for *Domain Similarity CBC*) in Figure 2. Overall, it applies the (modified) Phase I and II of CBC twice successively. The modifications to the original CBC are: (1') use the new similarity measure *dsSim* instead of cosine to compute the scores of the derived clusters; and (2') remove from committee centroids the features which co-occurred with less than σ percent of the member words or which are not in the top k features. All other parts of the algorithm are unchanged from CBC.

Note that DSCBC may still group "warm" and "cold" in the same cluster. However, such a cluster would have a lower *dsSim* value (hence the cluster score) because the domains of temperature and personality are not similar themselves, thus is less likely to be selected as a committee. If it were, features in less dominant domains (i.e., extended meanings) are discarded because only the top k features are retained in the centroid, thereby resulting in a monosemous committee. In this sense, a DSCBC committee is not equivalent to a synset – rather, it is a list of features (abstract nouns) which explicitly describe the semantic domain(s) of the

Step 1: Derive committees from data.
Apply Phase I and II of CBC to the dataset using:
- *dsSim* in computing avgsim(c)$^{(\mathbf{1}')}$
- the top k features that are also shared by more than σ percent of the words in the committee to represent its centroid.$^{(\mathbf{2}')}$

Step 2: Derive committees from committees.
Repeat Step 1, but with the committees derived in Step 1.

Step 3: Assign words to committees.
Apply Phase III of the CBC algorithm to the dataset using the final set of committees derived in Step 2.

Figure 2: The DSCBC Algorithm

cluster.

In summary, the key to overcome the polysemous cluster problem is to use externally obtained information about the similarity/dependency between features and incorporate the information in the clustering process.

5.3 Related Work

There are only few other works which attempted to automatically derive semantic categories of adjectives. Tanaka and Hirai (2001) used Latent Semantic Analysis (LSA) to cluster (Japanese) adjectives. However, they used individual senses of the adjectives recorded in a dictionary (*Kojien*) as the data instances, thus did not deal with the polysemy of adjectives.

Hatzivassiloglou and McKeown (1993) cluster adjectives by their properties or *scales*. For example, a set of adjectives {cold, cool, warm} is a linguistic scale which indicates a variation in the intensity of temperature. They extracted instances of adjective-noun pairs from a parsed corpus, and used Kendall's τ coefficient as the similarity measure and a hill-climbing partitive algorithm to cluster the extracted adjectives. However, their work focuses only on scalar adjectives, and does not concern groupings which indicate non-scalar properties (such as personality).

6 Evaluation

To evaluate our algorithm DSCBC, we applied it to the two datasets described earlier and compared the results with other clustering algorithms, in particular CBC, a graph-based clustering and K-means. For the graph and K-means algorithms, we used a tool called CLUTO.[7] Also for those two algorithms, we made some modifications so that they do soft-clustering (which assigns an instance to one or more clusters) in order to make them comparable with CBC and DSCBC. To make modifications, we closely followed the way described in (Pantel and Lin, 2002): first apply the algorithm to the dataset and obtain clusters, then apply *MK-means* (Zhong and Ghosh, 2003) using the centroids of those clusters as the initial centroids. MK-means is a generalized version of the standard K-means algorithm, and assigns each instance to one or more clusters with which it has the similarity greater than a pre-specified threshold. As with K-means, MK-means performs several iterations until a pre-specified number of iterations is reached. In our experiments, we set the maximum number of iterations to be 5, as with (Pantel and Lin, 2002).

Furthermore, since DSCBC is an extension of CBC with the domain feature similarity, we also implemented the versions of the graph and K-means algorithms which extend the base algorithms in the same way. So in all, we compared a total of six algorithms: DSCBC, CBC, DSGr, Gr, DSKmeans and Kmeans.

For the parameters in the DS extended algorithms (DSCBC, DSGr, DSKmeans), for the English dataset, we set $k = 10$ and 6, and $\sigma = 0.8$ and 0.6 in Step 1 and Step 2 respectively. We reduced k from Step 1 to Step 2 in order to refine the salient features. For σ, we used somewhat higher values, especially in Step 1, because the dataset was relatively dense (since word usage is extremely diverse in web pages). For the Japanese dataset, we set $k = 6$ in both steps, and $\sigma = 0.4$ and 0.2 in Step 1 and Step 2 respectively. We used the same k and lower σ values because the Japanese dataset was rather sparse. We decided on those parameter values based on our intuitions and quick preliminary inspections; they were not optimized for any specific algorithm tested.

6.1 Derived Committees

First we examined the committees derived by the algorithms. Table 1 shows some statistics. DSCBC produced 38 committees for both datasets (coincidentally), while CBC produced 35 and 50 committees for the Japanese and English

[7] http://glaros.dtc.umn.edu/gkhome/views/cluto

Table 1: Some Committee Statistics

	Total # comm.	Ave # feat.	Ave Domain Tightness (p-value)	Ave Cosine (p-value)
Japanese				
DSCBC	38	5.7	0.172 (–)	0.026 (–)
CBC	35	5.8	0.138 (*)	0.025 (0.447)
DSGr	38	6.0	0.153 (0.08)	0.069 (*)
Gr	35	6.0	0.154 (0.09)	0.056 (*)
DSKmeans	38	6.0	0.146 (*)	0.044 (*)
Kmeans	35	6.0	0.142 (*)	0.037 (*)
English				
DSCBC	38	3.3	0.470 (–)	0.010 (–)
CBC	50	4.9	0.196 (*)	0.012 (0.18)
DSGr	38	6.0	0.224 (*)	0.018 (*)
Gr	50	6.0	0.221 (*)	0.022 (*)
DSKmeans	38	6.0	0.209 (*)	0.012 (0.20)
Kmeans	50	6.0	0.198 (*)	0.011 (0.37)

dataset respectively. Other DS algorithms (DSGr and DSKmeans) and non-DS algorithms (Gr and Kmeans) were pre-specified to produce the same number of committees as DSCBC and CBC respectively. The average number of (non-zero) features was 5.7 and 3.3 by DSCBC, and 5.8 and 4.9 by CBC, for Japanese and English respectively. For all other algorithms, the average number of features was 6 (= k, as described in section 5.2).

The column "Ave Domain Tightness" indicates the average domain tightness (dt) of the committees derived by each algorithm. Here, a higher value means the feature domain similarity of the committees are overall more similar, therefore the committees are less polysemous. As you can see, DSCBC showed the highest tightness (0.172 and 0.470 respectively). The results by all other algorithms, including other DS algorithms, were much lower than DSCBC, and the differences were statistically significant, as evidenced by the p-values[8] (where a symbol (*) indicates < 0.05). As a note, the domain tightness was in general much lower for the Japanese dataset. It is probably due to the difference in the sources used to compute the feature domain similarity – for Japanese, the case frame relations were collected from the Web and the words were not disambiguated, so the data is somewhat noisy, whereas for English we used the first two senses of a word in *javasimlib*.

The column "Ave Cosine" indicates the average pairwise cosine between the committees for each algorithm. Here, a lower value means the committees are dissimilar to each other, thus scattered well in the feature space. The result shows that DSCBC had the lowest value, although the differences were not statistically significant in some cases. As for the graph and K-means algorithms, the graph in general seems to produce more monosemous but more correlated clusters than K-means, as observed for both datasets.

[8]The p-values shown in Table 1 are obtained by one-sided t-tests against DSCBC.

Japanese
{*shinjo* "feeling", *kimochi* "feeling", *kibun* "mood", *kanjo* "emotion", *omoi* "feeling"}
{*imiai* "meaning", *shikisai* "tint", *imi* "meaning", *sokumen* "side", *men* "side"}
{*kibo* "scale", *gaku* "amount", *ryo* "volume", *kazu* "quantity", *aida* "interval", *deido* "degree"}
{*kuuki* "air", *kikou* "climate", *tenki* "weather", *seikaku* "personality", *imeji* "image"}
English
{smell, aroma}
{appearance, look}
{quantity, amount, number}
{attitude, countenance, nature}
{day, shade, room, light, face, color}

Figure 3: Example DSCBC Committees

Figure 3 shows some example committees derived by DSCBC for both datasets. Most of the committees seem to pick out a single semantic domain fairly well, but some are still polysemous clusters. For example, {*kuuki* "air", *kikou* "climate", *tenki* "weather", *seikaku* "personality", *imeji* "image"} in Japanese is a mixture of weather and personality (as we have been using as an example in this paper). Similarly, {day, shade, room, light, face, color} in English seems to group different domains – probably caused by the polysemy of color or luminance adjectives such as "bright" and "dark".

6.2 Assigned Word Senses

Next we inspected the senses/committees assigned to words. For each word, the algorithms assign one or more committees. Here, each committee should correspond to a sense of the word. Figure 4 shows some examples of the assigned senses by DSCBC. To determine whether or not a committee indeed corresponds to a correct sense of a word, we compared it against the sense encoded in manually created lexicons as gold standards: *Daijirin*[9] for Japanese, and WordNet for English. Then the evaluation was measured with respect to precision and recall.

[9]The web version available at http://dictionary.goo.ne.jp/

Japanese
atatakai "warm"
{*kuuki* "air", *kikou* "climate", *tenki* "weather", *seikaku* "character", *imeji* "image"}
{*hitogara* "personality", *seishitsu* "nature", *monogoshi* "manner"}
yawarakai "soft"
{*kankaku* "sensation", *kanshoku* "touch", *aji* "taste", *yosa* "merit"}
{*hitogara* "personality", *seishitsu* "nature", *monogoshi* "manner"}
hiroi "wide"
{*kibo* "scale", *gaku* "amount", *ryo* "volume", *kazu* "quantity", *aida* "interval", *deido* "degree"}
{*shiya* "view", *kenchi* "viewpoint", *kanten* "viewpoint", *kakudo* "angle"}
English
"cold"
{complexion, color, face}
{attitude, countenance, nature}
"democratic"
{framework, concept, approach, idea, model}
{tendency, view, nature}

Figure 4: Example Sense Assignments by DSCBC

6.2.1 The English Dataset

For the English dataset, we first looked up each of the 1500 adjectives in the dataset in WordNet and obtained its senses/synsets. Then for each synset, we mapped it to its related noun so that we can utilize the WordNet noun hierarchy to compute the similarity/correspondence.[10] For a given synset, we traversed the *attribute* relation encoded in WordNet, for example, "warm#1" → "temperature#1". If the attribute relation was not available, we traversed the *derivationally related form* relation, for example, "warm#4" → "warmness#1". Word senses which are not indicated with either relation were ignored in the evaluation. The average number of senses (of the 1500 adjectives) which were associated with either relation was 1.63. Finally, we determined that an automatically derived committee corresponds to the mapped noun synset if the average similarity between the nouns in the committee and the synset is above a threshold, that is,

$$\frac{1}{|c|}\sum_{f\in F(c)} jsim(f,s) \geq \theta$$

where c is a committee, f is an abstract noun in $F(c)$ (the set of (non-zero) features in c), s is the mapped WordNet noun synset, $jsim(f,s)$ is the similarity between f and s obtained through *javasimlib*, and θ is the threshold. For $jsim(f,s)$, the top (at most) two senses were considered for the feature noun f, and the maximum of the two values returned by *javasimlib* ($jsim(f\#1,s)$ and $jsim(f\#2,s)$) was used.

Then we computed the *precision* of an adjective a as the ratio of the correctly assigned committees of all committees assigned by the algorithm for a. The precision of an algorithm was the average precision of all adjectives in the dataset.

The *recall* of an adjective a, on the other hand, was computed as the ratio of the correctly discovered senses of all senses encoded in WordNet for a. This recall actually is a tough measure, because the coverage of our dataset is much more limited than WordNet. But we thought this measure could provide some indication on the coverage of the algorithms and be utilized to rank the algorithms. The recall of the algorithm was the average recall of all adjectives in the dataset.

Table 2 shows the precision, recall and F-measure for the English dataset when θ = 0.25. The F-measure was computed as standard:

$$F = \frac{2RP}{R+P}$$

where R is the recall, P is the precision. As you can see, DSCBC produced the highest precision as well as recall. Also, the DS-extended algorithms performed considerably better than their non-DS counterparts for both precision and recall – verifying the positive effects made by the incorporation of domain feature similarity. Also notice CBC showed a better performance over other standard algorithms (graph and K-means), including their DS-extended versions. This verifies that CBC is indeed an effective tool for word sense discovery, and that using CBC as the base algorithm for extension in our work was a good choice.

One thing to note about the results is that precision is overall rather low. The main reason would be because we only examined a subset of the WordNet senses. When an assigned sense indeed corresponded to a WordNet sense but if it is not mapped to a noun synset in WordNet, the assigned sense ended up having no matches, thus ultimately was considered incorrect.

To investigate further, we also inspected the performance of the algorithms with different values of the threshold θ, since the determination of a correct assignment is dependent on this value. As θ is raised, the precision and recall (thus F-measure)

[10] In WordNet, adjectives are not organized hierarchically; instead they are simply categorized into two large groups (descriptive and relational adjectives).

Table 2: Precision, Recall and F-measure for the English Dataset (when $\theta = 0.25$)

	Precision	Recall	F-measure
DSCBC	0.397	0.397	0.397
CBC	0.304	0.264	0.283
DSGr	0.214	0.251	0.231
Gr	0.192	0.202	0.197
DSKmeans	0.254	0.213	0.232
Kmeans	0.169	0.162	0.166

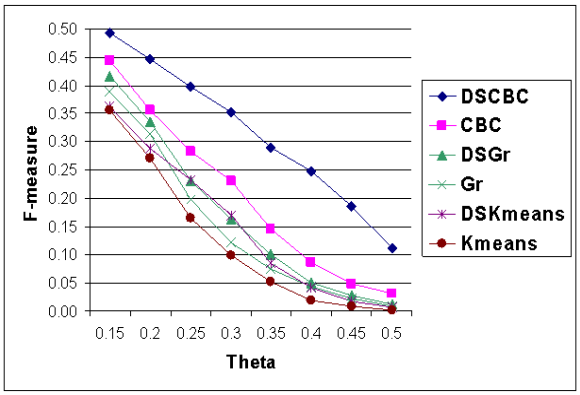


Figure 5: F-measure for varying θ for the English Dataset

will decrease, because only the higher correspondence values are considered as correct. Figure 5 shows the F-measure for varying θ for all algorithms. DSCBC has the highest F-measure consistently for all θ by a large margin. The DS algorithms are showing a better performance over their non-DS counterparts here as well for all θ, although the differences (between DSGr and Gr, and DSKmeans and Kmeans) are not as dramatic as the difference between DSCBC and CBC.

6.2.2 The Japanese Dataset

For the Japanese dataset, we were unable to do an automatic evaluation, because word senses in *Daijirin* are only enumerated with no particular organization. Instead, we randomly selected 60 words from the dataset, and manually compared the committees assigned by the algorithms against the senses encoded in *Daijirin*. The average number of senses of the selected 60 words was 4.05. For the algorithms, we focused on three algorithms: DSCBC and CBC, plus Gr which showed the tightest domain feature similarity among the non-CBC algorithms according to our inspection.

For each algorithm, we manually matched up the assigned committees to the word senses which we determined are correct. To judge the correctness, we looked at the top 3 features in a committee and determined correct if at least one of them corresponded to the sense based on our linguistic intuitions. Then we computed the precision as the ratio of the correctly assigned committees over all assignments (thus each sense assignment was taken individually). Similarly we computed the recall as the ratio of the correctly assigned committees over all word senses listed in the lexicon. Table 3 shows the results. As you can see, DSCBC produced the best result for this dataset as well.

Table 3: Precision, Recall and F-measure for the Japanese Dataset

	Precision	Recall	F-measure
DSCBC	0.571	0.576	0.574
CBC	0.532	0.440	0.482
Gr	0.590	0.449	0.511

Table 4: Average Ranks by Algorithm for the Japanese Dataset

	Recalled by All 3 Algo.	Recalled by At least 2 Algo.
DSCBC	1.302	1.190
CBC	2.206	1.638
Gr	2.492	1.802

To further investigate the quality of the derived committees, we also ranked the committees assigned to the same word sense by the three algorithms. Since not all word senses were discovered by all three algorithms, we looked at the ones that were recalled by all three and at least two algorithms. Table 4 shows the average ranks by the algorithms. Ranks were given in the descending order (from 1 to 3), so a lower value means a better correspondence to the lexicon senses. For both cases, DSCBC had the lowest rank, suggesting that DSCBC's committees were the closest to the gold standards. On the other hand, the quality of the committees derived by Gr is not as good as DSCBC or CBC – although Gr showed the highest precision, its ranks were the lowest.

7 Conclusions and Future Work

In this paper, we presented a new algorithm for discovering word senses for polysemous words, and showed the improved performance over other clustering algorithms. By incorporating the feature domain similarity, the algorithm produces clusters which are more "tight" with respect to the

domains, that is, more monosemous clusters.

For future work, an immediate task is to do a more thorough evaluation, manual as well as automatic, including an investigation on exactly which element(s) in the algorithm contributed the most to the improved performance. Our preliminary inspection indicates the feature domain similarity was the most significant factor. Finally, we would like to investigate the usefulness of the derived word senses in practical applications such as word sense disambiguation and question-answering.

References

G. Boleda, T. Badia, and E. Batlle. 2004. Acquisition of Semantic Classes for Adjectives from Distributional Evidence. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*.

T. Brants and A. Franz. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium. Catalog No. LDC2006T13, ISBN 1-58563-397-6.

V. Hatzivassiloglou and K. McKeown. 1993. Towards the Automatic Identification of Adjectival Scales: Clustering Adjectives According to Meaning. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL-93)*.

V. Hatzivassiloglou and J. Wiebe. 2000. Effects of Adjective Orientation and Gradability on Sentence Subjectivity. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*.

D. Hindle. 1990. Noun Classification from Predicate-argument Structures. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL-90)*.

J. Justeson and S. Katz. 1995. Principled Disambiguation: Discriminating Adjective Senses with Modified Nouns. *Computational Linguistics*, 21(1):1 – 27.

K. Kanzaki, E. Yamamoto, Q. Ma, and H. Isahara. 2004. Construction of an Objective Hierarchy of Abstract Concepts via Directional Similarity. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*.

D. Kawahara and S. Kurohashi. 2006. A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL-2006)*.

M. Lapata. 2001. A Corpus-based Account of Regular Polysemy: The Case of Context-sensitive Adjectives. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

D. Lin and P. Pantel. 2001. Induction of Semantic Classes from Natural Language Text. In *Proceedings of the 7th ACM Conference on Knowledge Discovery and Data Mining (KDD-01)*.

G. Miller. 1990. WordNet: An Online Lexical Database. *International Journal of Lexicography*, 3(4).

P. Pantel and D. Lin. 2002. Discovering Word Senses from Text. In *Proceedings of the 8th ACM Conference on Knowledge Discovery and Data Mining (KDD-02)*.

J. Pustejovsky. 1995. *The Generative Lexicon*. The MIT Press.

G. Salton and M. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

N. Seco, T. Veale, and J. Hayes. 2004. An Intrinsic Information Content Metric for Semantic Similarity in WordNet. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04)*.

T. Takahashi. 1975. Various Phases Related to the Part-whole Relationship Investigated in Sentences. *Studies in the Japanese Language*, 103:1–16. In Japanese.

M. Tanaka and Y. Hirai. 2001. Clustering Adjectives Based on Latent Semantic Analysis. Technical Report 2001-92, The Institute of Electronics, Information and Communication Engineers. In Japanese.

S. Walde. 2006. Experiments on the Automatic Induction of German Semantic Verb Classes. *Computational Linguistics*, 32(2):159 – 194.

J. Wiebe. 2000. Learning Subjective Adjectives from Corpora. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*.

S. Zhong and J. Ghosh. 2003. A Unified Framework for Model-based Clustering. *Journal of Machine Learning Research*, 4:1001–1037.

Building a clustered semantic network for an Entire Large Dictionary of Japanese

Hiroyuki Akama
Tokyo Institute of Technology, 2-12-1, O-okayama, Meguro-ku、JAPAN
akama.h.aa @ m.titech.ac.jp

Jaeyoung Jung
Tokyo Institute of Technology, 2-12-1, O-okayama, Meguro-ku、JAPAN
jung.j.aa@m.titech.ac.jp

Maki Miyake
Osaka University, 1-8 Machikaneyama-cho, Toyonaka-shi, Osaka, 560-0043, Japan
mmiyake@lang.osaka-u.ac.jp

Kenji Yosihasi
Tokyo Institute of Technology, 2-12-1, O-okayama, Meguro-ku、JAPAN
yoshihashi.k.aa@m.titech.ac.jp

Kikuko Nishina
Tokyo Institute of Technology, 2-12-1, O-okayama, Meguro-ku、JAPAN
nishina.k.aa@m.titech.ac.jp

Abstract

In this paper, we develop a new method of solving the cluster-size imbalance problem observed when documents and corpora are processed with the Markov Clustering (MCL) Algorithm (Van Dongen, 2000), which is widely recognized as an efficient approach to graph clustering. The Branching MCL (BMCL) and its variant, Reverse Branching MCL (RBMCL) can resize overly inclusive Markov clusters (core clusters) into appropriate subsets. The method, together with the Recurrent MCL (RMCL) that integrates concepts of similar words (Jung, 2006), is applied to a semantic network built from the large-scale corpus of Gakken's Large Dictionary of Japanese (GLDJ), covering 100,000 words, definitions, examples, and grammatical explanations. The effectiveness of these techniques is currently being tested by creating a clustered semantic network for the GLDJ.

1 Introduction

Large-scale semantic networks can be built by selecting 'word pairs' that share a lexical relationship, such as adjacency, association or co-occurrence. Within such networks, words and their relations are represented by vertices and edges forming a dense graph, which can have the appearance of distant clouds when viewed on a display. Graph clustering methods then allow us to break the graph down into clusters consisting of similar words that can be regarded as representing a 'concept'. For example, by alternating between the two steps of expansion (random walk) and inflation (Γgamma operation) to reach the convergence of a stochastic matrix, the Markov Cluster Algorithm (MCL) proposed by Van Dongen (2000) divides a graph into 'hard' clusters that have no mutual overlap.

The MCL process basically consists of calculating probabilities for the constant movement of a random walker until a probability value is converged on. In creating a data cluster, word pairs are initially selected from linguistic data. The word pairs are then used to create an adjacency matrix, which simply represents the binary relations within the data, and is, in turn, transformed into a transition matrix.

Based on an initial probability value, a random walker starts to move, and at each step, probability values are re-calculated for the next step. Thus, a probability is obtained at each step in the random walk as a product of powering the previous transition matrix (called the expansion calculation step) and of squaring each value for the transition matrix by Hadamard power and re-scaling (called the inflation calculation step). As these two calculation steps are repeated, the random walker becomes trapped at a certain point within a particular area because the probably value almost ceases to vary (converged status). At this stage, the graph finally is divided so there is no overlap (hard clustering).

Recently, MCL has been applied to various domains (Enright, et. al., 2002; Gfeller, 2005;

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 308–316,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

Dorow, et. al., 2005). However, when applied to corpus data, while MCL is effective in establishing order within a semantic network, there is a problem with the lack of proportion in cluster sizes. As the distribution of words in a document follows Zipf's law, with such networks, there is an imbalance in the distribution of cluster sizes, with either exceptionally large Markov clusters that lack a particular feature or with many isolated clusters consisting of a single word. This kind of imbalance is far from satisfactory when seeking to determine classified word groups (concepts) at a taxonomical level.

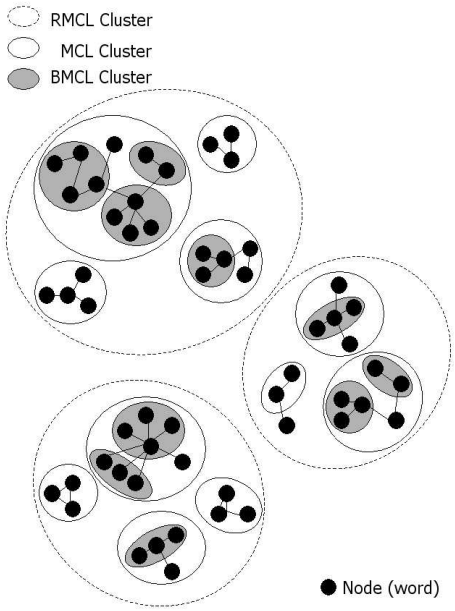


Figure 1 MCL, RMCL & BMCL

To overcome this problem to building an appropriate semantic network for a document, Jung et al. (2006) proposed an original graph clustering algorithm to refine the MCL: Recurrent MCL (RMCL), which allows for the reconnection of atomized Markov clusters by creating adjacency relationships among 'concept' clusters by regarding them as nodes. In this paper we deal with another new algorithm called Branching MCL (BMCL), which involves dividing large Markov clusters (referred to as core clusters here) into branches and redefining adjacency relationships among the resultant sub-clusters. We apply these size-control methods to the Markov clusters created in a semantic network representation of Gakken's Large Dictionary of Japanese containing approximately 100,000 words, their definitions, examples, and grammatical explanations.

The purpose of this work is to automatically build an exhaustive thesaurus or an associative concepts dictionary from a large scale dictionary or encyclopedia. The thesaurus produced in this way will enable web users to easily search documents on difficult topics by presenting them with some candidate synonyms for their query terms. Certainly, no information retrieval system yet exists that allows us to systematically learn the huge collection of vocabulary gathered together in the grand dictionary compiled by the renowned scholar Kindaichi. This work will also be beneficial in the creation of other types of dictionaries, such as associative concept dictionaries (such as Nelson's, EAT's and Ishizaki's examples), that consist of semantically-related word pairs with coefficients for distance and similarity based on lexical association task results. Many of these word pairing cannot be identified from traditional dictionary entries, but they emerge as products of our graph clustering methods.

2 Clustered Semantic Network of the Japanese Dictionary

Gakken's Large Dictionary of Japanese (GLDJ) (Kindaichi & Ikeda,1988) is an authoritative dictionary with some features of an encyclopedia in terms of the rich explanatory texts and copious examples. The first step in building the semantic network for GLDJ was to exhaustively enumerate instances of the headwords in the text (including examples) as part of a lexical entry. We then extracted all word pairs to make an adjacency matrix for the semantic network, by combining every headword with every other headword included within an entry text. The possibilities of generating relatively loose associations were introduced by incorporating some non-relevant words used in the example sentences. We also included in the word pairings metadata terms, such as "archaic", "literary", "animal", "plant", and "law", that provide generic information, even though the high frequencies of such terms tend to produce extremely large Markov clusters.

Thus, a 98,083* 98,083 adjacency matrix was calculated after removing noise words, functional words, and 1,321 isolated words that would seem to reflect limitations encountered with all Japanese morpheme analysis systems (particularly, the difficulties of handling homonyms written in hiragana). The MCL algorithm was then applied to the whole GLDJ semantic network in order to automatically create a thesaurus list of grouped similar words. This process finally formed an idempotent stochastic matrix at the 20th cluster

stage yielding 13,961 hard clusters, including all word nodes without overlap. The size (the number of words contained) of the clusters ranged from 1 to 8,730.

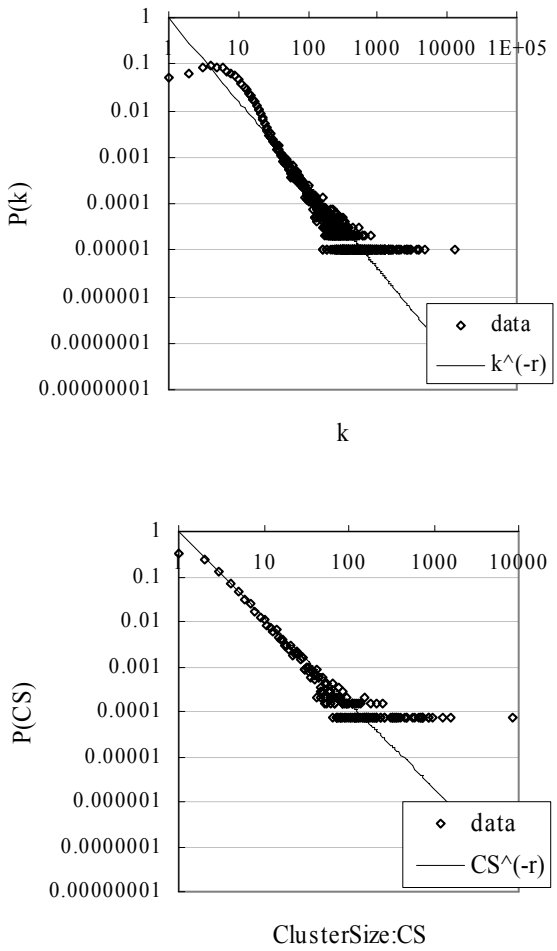


Figure2 Distributions for Degree and Cluster Sizes

Figure 2 contrasts the probability distributions for word degree k (upper) and Markov cluster sizes CS (lower). Both distributions have similar shapes and approximately follow power laws (r = 1.9 and r = 1.8, respectively). A striking feature that is common to both graphs is that the ranges in distribution of appearance probabilities p(k) extend to the right at the lowest values, highlighting the presence of some words that have extremely high degrees (hub nodes) and of some extremely large core clusters.

The MCL results indicate that words are generally appropriately divided into concepts. That is, every Markov cluster consists of word nodes with either high or moderate levels of similarity. While subjective evaluation of the initial MCL output is still on-going, we focus here on the RMCL and BMCL results to verify the effectiveness of these methods.

3 Branching MCL (BMCL)

The results of applying MCL to the semantic network indicate that all words were divided according to concepts corresponding to the respective Markov clusters. The next step in adjusting the clustered semantic network is to identify the relations between hard clusters (word groups) in order to reconnect them. The Recurrent Markov Cluster Algorithm (RMCL) proposed by Jung et al (2006) involves reversing the MCL process and retracing the connection history for hard clusters (word groups). While the clusters in the final stage have no words that overlap with other clusters, there are words that are common to more than one cluster during the clustering stages (called soft clustering stages). RMCL combines each prior MCL clustering stage with the resultant clustered stage. By obtaining adjacency information for these combinations, a new virtual graph is generated where each hard cluster (word group) is regarded as a node and is connected together. MCL is then repeatedly applied to this graph with a downsized adjacency matrix to create a classification of concepts. When applied to small-sized clusters of similar words, RMCL is an effective way of forming the clusters into higher level concept clusters by re-integration.

In contrast, our new algorithm of Branching Markov Cluster Algorithm (BMCL) divides extremely large Markov clusters (core clusters) into appropriate subsets. BMCL is a method of building adjacency relationships 'inside' MCL clusters, as opposed to the RMCL which operates between MCL clusters. In MCL, the vertices of a hard cluster at the final clustering stage are often crystallized into a simple star form. However, BMCL generates and rearticulates distinct subcategories within a wide subject that are linked together in a core cluster. With the BMCL, we extract from the original graph of raw data all paths of length 2 which are subsequently connected to any two vertices of a core cluster via a hub node extracted from other Markov clusters. A hub node is considered as the most significant word representing a concept cluster. The representative word has the largest degree of all words within the cluster.

The basic logic of BMCL involves applying the preprocessing steps for a second MCL to core

clusters only. If there is a connection between the vertices i,k, we represent that 'real' adjacency by $d(i,k)=1$. However, it is most important for the BMCL to compute what we refer to as the 'latent' adjacency $d_v(i,j)=1$, which can be defined as follows. Let the hub M_h of each Markov cluster M be the vertex with the largest degree of M. Here, we set a sufficiently large core cluster C, a set of hubs H and the hub of C as C_h.

Under such conditions, the set of outside hubs bypassing the intra-core connections $K_{i,j}$ can be formulized as;

$\{K_{i,j} \mid K_{i,j} \subset H, k \in K_{i,j}, d(i,k)=d(j,k)=1\}$,

where $i,j \in \underset{i \neq j, i \neq C_h, j \neq C_h}{C}$, $C_h \notin H$. Moreover, we have developed an original $\underset{n}{ArgTopn}$ function, which identifies the set of n nodes that have the highest connection values.

In order to realize a moderate connection rate that can support appropriate MCL operations, two kinds of pruning thresholds, θ_p, θ_q, are set in the row direction by fixing i to the number of the shortest paths between i,j -- $|K_{i,j}|$ -- to make a latent adjacency matrix for the core cluster according to the following connection rule.

$$if(|K_{i,j}| \geq \theta_p \,\&\&\, j \in \underset{n=\theta_q}{ArgTopn} \mid K_{i,j} \mid)$$

$$-> d_v(i,j)=1$$

MCL is applied once again, but this time to the latent adjacency matrix which allows us to obtain from the appropriately resized sub-clusters a clear and understandable series of subordinate categories.

Figure 3 shows how a core cluster is divided into appropriately-sized sub-clusters when BMCL is applied to the GLDJ data. The criterion for regarding a cluster as a 'core' cluster is whether its size is two or more standard deviations above the mean. An important feature of BMCL is how it avoids the risk of over-connection due to these detours and which can block further subdivisions in subsequent MCL operations. Therefore, it is better to remove a representative node (hub) from the member node list for a core cluster (for example, in the case of the GLDJ core cluster, as seen in Figure 3, つくる_作る_造る (/tsukuru/, make) is the hub and 代用品 (substitute), 贋 (imitation), 贋札 (counterfeit bill), 局紙 (fine quality paper), 地下室 (basement), 屋根裏 (garret), 嘘泣き (pretend to be crying) and 泣き真似 (feign tears) are some of the member nodes). Such connections made by intermediary exterior hubs (紙幣 (paper money), 本物 (real thing), 使う (use), 偽る (falsify), 泣く (cry), 直ぐ (immediately), and 部屋 (room)) form a latent adjacency between two semantically similar words in a core cluster to replace an edge that was lost in the MCL. When we execute MCL on this latent adjacency matrix for inside a core cluster (apart from the hub), then the core cluster is divided into smaller Markov clusters that each represent the respective internal concepts (counterfeit money, false cry and hidden rooms).

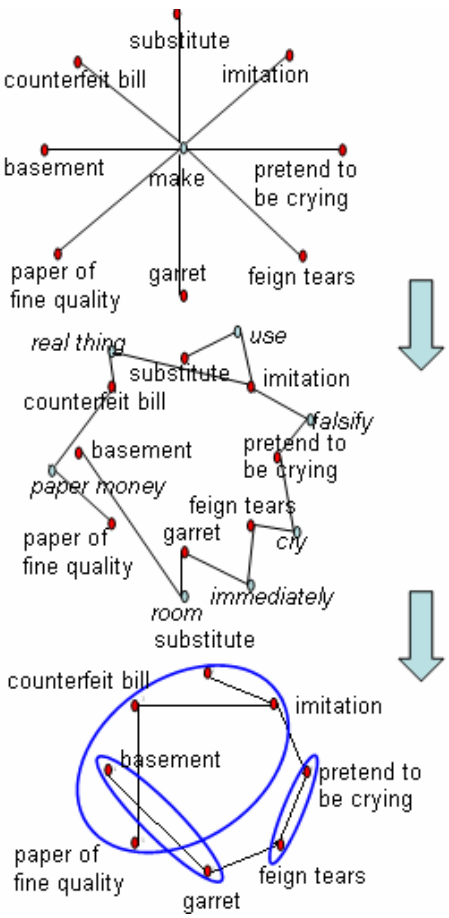


Figure 3: The Mechanism of BMCL

The following formula illustrates the steps in the RMCL and BMCL algorithms.

#The MCL algorithm follows but slightly modifies Figure 15 presented in Van Dongen's thesis (p.55).

MCL (G,e,r) {

$G=G+I; T_1=T_G;$

for k=1,...,∞{

$T_{2k}=\mathrm{Exp}_e(T_{2k-1})$; $T_{2k+1}=\Gamma r(T_{2k})$;

#Expansion and Inflation.
#Initial cluster stage.
for i=1,...,n {
T_{2k+1} = = [t_{ij}](i=1,2,…,m; j=1,2,…,m);
C_i={[t_{ij}]| for j=1,...,m{[t_{ij}]>0.1};};
}
#Final cluster stage.
ClusterStage$k$={C_k(1), C_k(2), ..., C_k(d)};
If(T_{2k+1} is (near-) idempotent) break;
}
#The following formula is for our original algorithm BMCL to divide large core clusters.
#Select Core Clusters by their size, for example,
if (Size(C_k(p)) > 2 × Standard Deviation from Mean(Size C_k(j)), CoreCluster$n$ = C_k(p);
# Remove the representative node (hub node) for the core cluster from the following two lists which are arguments of the Complement function,
CC'n= Complement(CoreCluster$n$, Representaive_ClusterStage$k$)
RCS'k= Complement (Representaive_ClusterStage$k$, CoreCluster$n$)
#ExtractAdjacency2(adjacency_matrix, {row_number_list, column_number_list}) is to extract the rows and columns of an adjacency matrix. G is the original adjacency matrix.
CC'n_RCS'k = ExtractAdjacency2(G, {CC'n, RCS'k })
PathNumbersMatrix$n$ = CC' n_RCS'k × Tr(CC' n_RCS'k);
#Generate an adjacency matrix by setting all diagonal elements = 0 and pruning non-diagonal elements so that remaining ones = 1. The two pruning thresholds are described in this section.
LatentAdjacencyMatrix$n$ =MakeAdjacencyMatrix (PathNumbersMatrix$n$);
#Repeat MCL,
MCL(LatentAdjacencyMatrix$n$);

Figure 4 is a simplified illustration of the BMCL which involves making new connections among core cluster words. In order to generate sufficient numbers of links and to identify suitable connections between words, we use neighboring nodes placed outside of the core cluster. Considering an optimal clustering, the hub of the core cluster is removed in order to focus on the external connectors used to bridge any two internal words. Latent adjacency information can be taken selectively by counting the number of new links, which implies the extent of coupling strength. Finally, we apply the MCL once more to the latent adjacency matrix to appropriately subdivide the core cluster.

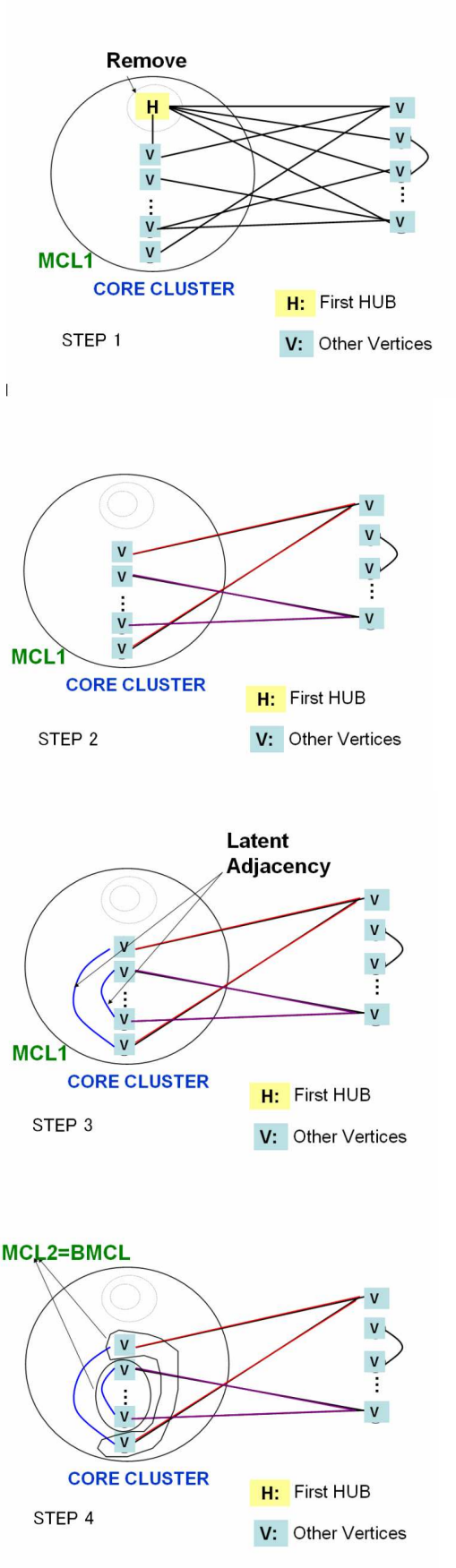


Figure 4 Simplified Mechanism of BMCL

4 BMCL Results

With the threshold $\theta_p = 1, \theta_q = 3$ (the three largest values in each row of the PathNumbersMatrix), the MCL generated nine core clusters for the GLDJ, and each of them was treated as an object for the BMCL (Table 1). We weighed the recall rate in order to produce as many core clusters as possible from the semantic network, by automatically and exhaustively treating the most general words that are characterized as hubs by their rich variety of meanings. Accordingly, we made no distinctions between ontological and semantic category information (ぶんご_文語 (/bungo/, archaic words), どうぶつ_動物 (/doubutsu/, animal), しょくぶつ_植物 (/syokubutsu/, plants), domain information (ぶっきょう_仏教 (/bukkyou/, Buddhism), りがく_理学 (/rigaku/, science), ほうりつ_法律 (/houritsu/, law)), and collocations (ようす_様子_容子 (/yousu/, appearance), つくる_作る_造る (/tsukuru/, make), ひとつ_一つ(/hitotsu/, one)).

However, we have carried a semantic analysis for just the fifth core cluster つくる_作る_造る (/tsukuru/, make), where all the member words are relatively easy for Japanese linguistic researchers to evaluate compared to the highly-technical words included in other core clusters that represent areas of expertise.

Despite executing the BMCL, the cluster-size imbalance problem still remains with MCL. Setting aside some (sub) core clusters and clusters consisting of a single word, we evaluated the remaining groups of moderate size. Google searches were employed to roughly estimate the closeness of two paired words extracted from the BMCL clusters for the fifth core cluster つくる_作る_造る (/tsukuru/, make).

Table 1 Representative Nodes for Core Clusters

Num	Representative Node	Size
1	ぶんご_文語 (archaic words)	8962
2	ようす_様子_容子 (appearance)	1651
3	ぶっきょう_仏教 (Buddhism)	1281
4	りがく_理学 (science)	947
5	つくる_作る_造る (make)	879
6	しょくぶつ_植物 (plants)	862
7	どうぶつ_動物 (animal)	715
8	ほうりつ_法律 (law)	703
9	ひとつ_一つ(one)	700

Our motivation for resorting to such an evaluation method is related to our future objective to automatically produce a *large-scaled associative concept dictionary* consisting of word pairs with their similarity coefficients. Here, we focus only on BMCL clusters containing two or three words to sample all pairs for research queries by inserting double quotation marks around each word and taking into account written variants due to combinations of hiragana, katakana and kanji. Figure 5 presents Google search results for 50 word pairings plotting word pair numbers across a series of hit ranges. Even though the presence of some rare, classic words generated extremely low recalls (for example, there were no hits for the pair of synonyms 急ぎ物 (/isogimono/) and 急造 /kyuzo/), the results indicate a degree of accuracy for our BMCL method.

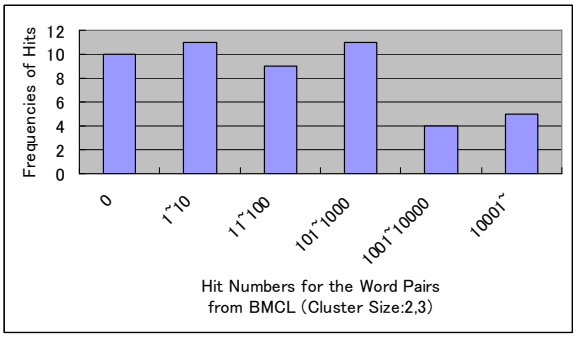


Figure 5 Google search results
y-axis = number of word pairs
x-axis = hit ranges

5 Reverse BMCL (RBMCL)

If a latent connection is counted, instead of paths of length 2 traced in the original adjacency, as a detour via a vertex of another cluster, it is possible to reverse the direction of network flow and to rebuild adjacency relationships 'outside' the core cluster, by connecting the hubs of other Markov clusters mediated through some core cluster members. In other words, switching within the BMCL algorithm the positions of the two matrices on the right-hand side of the formula for computing relationships *in* and *out* of a core cluster CC'*n* would yield the following *k***k* matrix, which provides the number of paths of length two between any two MCL representative nodes (RCS'*k* at the cluster stage *k*) that can be connected by way of one or more members of the CC'*n*.

PathNumbersMatrix$k$ = Tr(CC' n_RCS'k) × CC' n_RCS'k

This effectively totally reverses the BMCL algorithm at the starting point, while preserving all the formulas apart from the one that counts network paths to generate another latent adjacency matrix, particularly for the exterior hubs related to the very concept of the core cluster. Thus these outside representative nodes whose reference group is precisely the core cluster are clustered by the MCL when it is applied to the matrix. This method (which we call Reverse MCL or RBMCL) yields results similar to those of RMCL, but the clustering of representative nodes is executed exclusively from the perspective of the core concept. So, the RBMCL allows us to classify the connotative meanings of the core cluster, even though they have been excluded from it and have become dispersed and autonomic. Moreover, as the members of the RBMCL are all hub nodes, the clustering results are relatively easy to interpret due to the generality of their meanings or their polysemy.

The RBMCL applied to the fifth core cluster つくる_作る_造る (/tsukuru/, make) produced 940 clusters with a length mean and a standard deviation of 6.92 and 23.79, respectively. Google research evaluation results, conducted under the same conditions as for the BMCL, indicated that 58% of the word pairs obtained by RBMCL had more than 100,000 hits and 37% had been between 10,000 and 100,000 hits. It is worth noting that for some of the larger RBMCL clusters, multiple instances were found in Web documents, such as {貯蔵 (storage), アルコール (alcohol), 洋酒 (whisky and wine), 蒸留 (distillation), 産 (product), 年間 (year), 樽 (barrel)}, {運命 (fate), 勝つ (win), 彼の (concerned), 試合 (match), 勝敗 (victory or defeat), 戦い (battle), 戦う (fight), 拙い (unskillful), 武人 (warrior), 負ける (lose)} and {ゴム (rubber), 防水 (waterproofing), 色 (color), 鉛筆 (pencil), インキ (ink), 薬品 (medicine), 筒先 (tip_nozzle)}.

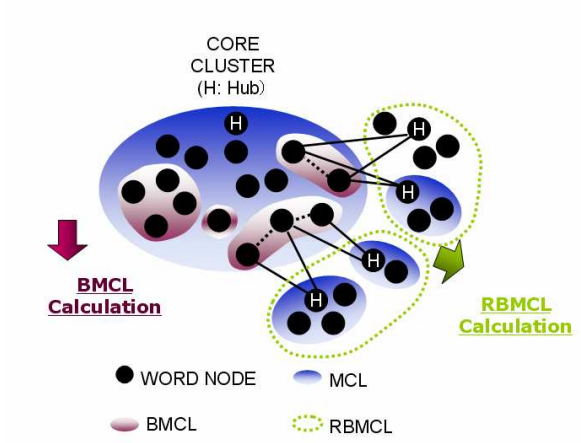


Figure 6 BMCL & RBMCL

6 Hierarchical Graph Clustering Methods and Their Evaluations

As noted earlier, the aim of this study is to automatically build an exhaustive thesaurus from a large-scale dictionary using a graph clustering method. However, it is quite difficult to evaluate this particular goal because the obtained results are unsupervised word classifications that can only be judged authoritatively by domain specialists (especially in terms of concept size and class determination of polysemic words). From the perspective of graph-clustering algorithms, it is also undeniable that, while real data is crucial for method testing, appropriate data is extremely rare (Delling et al., 2006). In short, we must acknowledge the limitations of the semantic approach for evaluating the outputs of large-scaled and complex systems.

The only criterion for directly evaluating the subdivisions of a whole graph into reasonably-sized partitions is the network *modularity* value (Newman and Girvan, 2004), which is used for both *detection* and *assessment* of clustered communities within networks. Modularity Q indicates differences in edge distributions between a graph of meaningful partitions and a random graph under the same vertices conditions (the numbers and the sum of their degrees). According to Newman et al., Q is defined as;

$$Q = \sum_i (e_{ii} - a_i^2),$$

where i is the cluster number of cluster c_i, e_{ii} is the proportion of internal links in the whole graph and a_i is the expected proportion of c_i's edges calculated as the total number of degrees in c_i divided by the total of all the degrees (2*the number of all edges) in the whole graph.

It is well known that modularity Q has become an extremely key concept for graph clustering algorithms that do not use spectral analysis as a core method. Newman's clustering algorithm, for example, which is now widely regarded as providing a baseline for all agglomerative hierarchical methods, consists of starting from singleton clusters and merging clusters at each dendrogram generation step by adopting a "greedy" strategy for modularity maximization. In other words, modularity Q is computed when selecting branches to be combined at each level of the tree where clusters are agglomerated in a step-wise fashion. However, it is 'reused' to decide at which point of the tree is it possible to crosscut in order to obtain the best partition of the network. Thus, modularity for Newman and those customizing his methods always searches for the best choices when cluster formation is still in progress and, even when it is already over and one looks back on the process.

Incidentally, our new BMCL combined with the RBMCL can be regarded as an approach to customizing algorithms in order to incorporate *hierarchical structure* into the Markov Clustering method, which, in its original form, does not generate any partial subsets. However, although level settings have nothing to do with dendrogram formation by splitting or merging clusters, in order to evaluate partitioning effects supplemented to MCL, we employ modularity Q as a helpful measure of clustering results, even though it is essentially not advantageous for the MCL which is not based on a modularity optimization algorithm.

It is certainly true that the size adjustments and the hierarchical structure realized for some graph areas through BMCL and RBMCL render the meanings that they represent more relevant and understandable. However, we now need to know whether these algorithms profoundly affect the accuracy of the whole clustering space by examining changes in the overall modularity Q. If, when compared to MCL baseline values, modularity Q values for the BMCL or the RBMCL results increase, or, at least, do not decrease abruptly, one would be able to conclude that semantic optimization by these methods ensures fitness for the overall clustering readjustment.

In terms of the DLDJ, the original division made by MCL resulted in a low modularity Q value of 0.1756. As for the BMCL results, the modularity Q value for the whole graph with BMCL clusters (rather than core cluster 1) was 0.1595. Modularity calculated solely for the BMCL clusters increased to 0.2214 from 0 (i.e., the modularity of core cluster 1 considered as a total graph). From these results, we can conclude that BMCL do not alter the overall coordination produced by the original MCL.

If we compare the BMCL and RBMCL results with results from ordinary sub-graph clustering methods, we can find some points of interest. Eliminating the hub from a core cluster yields a less dense adjacency matrix, as a lot of the nodes that dangle from the hub become isolated. When the MCL is applied to the original local matrix it produces a clustering result of remarkably high accuracy (0.769 without the hub compared to 0.103 with it), although there is very low recall (just 37% in the case of the GLDJ as core members are lost through isolation). In contrast, BMCL guarantees relatively high recall (88%) with latent adjacency effects retaining the lowest degree vertices, even though precision (as measured in terms of modularity Q) is relatively modest. The merits of BMCL are, however, evident in its counterpart to RBMCL clustering, because modularity Q value for the RBMCL clusters using the latent adjacency matrix is 0.2132, although connection between the RBMCL hubs in the original adjacency matrix records a very low modularity Q of 0.0827 when this partial matrix is directly submitted to MCL without using latent adjacency computation for the same area.

7 Future Works

Comprehensive evaluations of the results from MCL, RMCL, BMCL and RBMCL for the GLDJ data are beyond the scope of this paper due to the vast amounts of time that would be required for non-automatic processing of the numerous rare words within the data. Moreover, we accept that there is still room for improving the details of the algorithms, particularly, in terms of the data-dependent logic in settling latent adjacency for BMCL. Nevertheless, our new graph clustering methods are clearly appealing in terms of their simplicity as approaches to controlling cluster sizes which is crucial in building semantic networks from large-scale document data. Despite shortages of manpower to check and refine the output, we are currently developing a hierarchically-structured thesaurus based on the complete GLDJ, together with an Internet viewer system.

Acknowledgements

The research reported in this paper has been mainly supported by grants from the 21st Century Center of Excellence Program "Framework for Systematization and Application of Large-scale Knowledge Resources" (COE21-LKR, leader: Professor Sadaoki Furui). We would like to acknowledge the generosity of GAKKEN CO., LTD., in donating to the COE21-LKR the text file for the Large Dictionary of Japanese used in our work.

Reference

Clauset, A, Newman M.E.J., and Moore, C. Finding Community Structure in Very Large Networks, Phys. Rev. E 70, 066111 (2004)

Danon, L., Diaz-Guilera, A., and Arenas, A. Effect of Size Heterogeneity on Community Identification in Complex Networks, J. Stat. Mech. P11010 (2006)

Delling, D., Gaertler, M., Gorke, R., Nikoloski, Z., and Wagner, D.How to Evaluate Clustering Techniques, Technical Report 2006-24, ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), (2006)

Dorow, B. et al. Using Curvature and Markov Clustering in Graphs for Lexical Acquisition and Word Sence Discrimination, MEANING-2005,2nd Workshop organized by the MEANING Project, February,3rd-4th.(2005)

Enright, A. J. et al. An efficient algorithm for large-scale detection of protein families, Nucleic Acids, 1;30(7), pp.1575-84. (2002)

Gfeller, D. et al. Synonym Dictionary Improvement through Markov Clustering and Clustering Stability, International Symposium on Applied Stochastic Models and Data Analysis, pp.106-113. (2005)

Jung, J., Miyake, M., Akama, H. Recurrent Markov Cluster (RMCL)--Algorithm for the Refinement of the Semantic Network, LREC2006(International Conference on Language Resources and Evaluation), pp.1428-1432. (2006)

Jung, J., Miyake, M., Akama, H. Markov Cluster Shortest Path Founded upon the Alibi-breaking Algorithm, CICLing-2006, LNCS 3878, Springer Verlag Berlin Heidelberg, pp.55-58. (2006)

Kindaichi, H., Ikeda, Y. Gakken's Large Dictionary of Japanese, GAKKEN CO, LTD. (1988)

Kiss, G.R., Armstrong, C., Milroy, R., and Piper, J. An associative thesaurus of English and its computer analysis. In Aitken, A.J., Bailey, R.W. and Hamilton-Smith, N. (Eds.), The Computer and Literary Studies. Edinburgh: University Press, (1973)

Nelson, D.L., McEvoy, C. L. and Screiber, T.A. The University of South Florida word association, rhyme, and word fragment norms, http://www.usf.edu/FreeAssociation (1998)

Newman M.E.J., Detecting Community Structure in Networks, The European Physical Journal B Condensed Matter and Complex Systems Publisher Springer, pp.321-330. (2004)

Newman M. E. J. and Girvan M., Finding and evaluating community structure in networks, Physical Review E 69. 026113, (2004)

Okamoto, J., Ishizaki, S. Associative Concept Dictionary and its Comparison with Electronic Concept Dictionaries, http://afnlp.org/pacling2001/pdf/okamoto.pdf, (2001).

Tasgin, M. Community Detection model Using Genetic Algorithm in Complex Networks and its Application in Real-Life Networks, B.Sc. in Computer Engineering,Middle East Technical University, (2002)

Van Dongen, S. Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht. (2000)

A Named Entity Recogniser for Question Answering

Menno van Zaanen and **Diego Mollá**
Centre for Language Technology
Macquarie University
Sydney
Australia
{menno, diego}@ics.mq.edu.au

Abstract

Named Entity Recognisers (NERs) are typically used by question answering (QA) systems as means to preselect answer candidates. However, there has not been much work on the formal assessment of the use of NERs for QA nor on their optimal parameters. In this paper we investigate the main characteristics of a NER for QA. The results show that it is important to maintain high recall to retain all possible answers on the one hand, while high precision is essential during the final answer selection phase. We present an NER designed for QA, which aims at having a high recall.

1 Introduction

Named Entity (NE) recognition is the task of finding instances of specific types of entities in free text and is performed by a Named Entity Recogniser (NER). Typical entity types are person or company names, dates and times, and distances. Firstly defined as a separate task in the Message Understanding Conferences (Sundheim, 1995), NE recognition is currently used in a varied range of applications, such as bioinformatics, the identification of entities in molecular biology (Humphreys et al., 2000), and text classification (Armour et al., 2005).

In this paper we will focus on the use of NE recognition for text-based question answering (question answering or QA henceforth). There has been a major increase of research in QA since the creation of the question answering track of TREC (Voorhees, 1999), and nowadays we are starting to see the introduction of question-answering techniques in mainstream web search engines such as Google[1], Yahoo![2] and Microsoft[3].

A major part of the current research in QA focuses on finding fact-based answers. Often answers to these factoid questions are NEs. For this reason most QA systems incorporate a NER to detect answer candidates that are processed by subsequent modules. The positive impact of NE recognition in QA is widely acknowledged and there are studies that confirm this (Noguera et al., 2005, for example). However, there is no formal study of the optimal characteristics of a NER within the context of QA. Most often a NER is used off-the-shelf, without any fine tuning and usually it is treated as a black box that has been developed independently of the task of QA.

This paper is one step towards a formal study of the impact of NEs in a QA system. In particular, section 2 comments on the desiderata of a NER for QA. Next, section 3 describes the QA framework on which a selection of NERs will be evaluated. Section 4 introduces the NERs under evaluation, with special emphasis on AFNER, a NER designed specifically for QA. Section 5 presents the results of various experiments evaluating the NERs, and finally section 6 presents the concluding remarks and lines of further research.

2 Named Entity Recognition for Question Answering

Within the context of QA, a NER is used in two ways: to filter out strings (such as sentences) that do not possibly contain the answer, or to find possible exact answers. In the first case, the type of the expected answer is determined during the question analysis stage and this type is mapped to a list of named entity types. The NER is then used to single out the named entity types appearing in a text fragment. If a piece of text does not contain

[1] http://www.google.com
[2] http://search.yahoo.com
[3] http://search.msn.com

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 317–324,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

any named entity with a type compatible with the type of the expected answer, the text is discarded or heavily penalised.

In the second case, the NER is applied to a string and the found NEs are considered to be possible answers. Once all NEs are found, the expected answer type that was found during the question analysis phase (together with other information) is used to select the NE that best fits the question. This is the case we will be mainly focussing on in this article.

A NER developed for a generic NE recognition task (or for information extraction) is typically fine-tuned for a good balance between recall and precision. In this paper, however, we investigate whether this is what is needed in the context of QA. We expect that for QA a NER that concentrates on high recall and lower precision performs better than one with high precision but low recall. The rationale is that with high recall, even though more incorrect potential answers are passed on to the next phase after NER, the score of these wrong answers can be lowered using other means in subsequent modules. However, if a correct answer does not pass the NER filter (which happens with low recall) the answer will never be found. The results shown in this article, however, illustrate that it is not as simple as this.

To measure the impact of recall in a QA context, we present a newly developed NER that concentrates on providing recall that is higher than that of other freely available NERs. We will then test the impact of recall of NERs in a simple QA system.

3 Question Answering

The experiments discussed in this paper have been conducted within the AnswerFinder project (Mollá and van Zaanen, 2005). In this project, we develop the AnswerFinder question answering system, concentrating on using shallow representations of meaning to reduce the impact of paraphrases (different wordings of the same information). Here, we report on a sub-problem we tackled within this project, the actual finding of correct answers in the text.

The AnswerFinder question answering system consists of several phases that essentially work in a sequential manner. Each phase reduces the amount of data the system has to handle from then on. The advantage of this approach is that progressive phases can perform more computationally expensive operations on the data.

The first phase is a document retrieval phase that finds documents relevant to the question. This greatly reduces the amount of text that need to be handled in subsequent steps. Only the best *n* documents are used from this point on.

Next is the sentence selection phase. From the relevant documents found by the first phase, all sentences are scored against the question. The most relevant sentences according to this score are kept for further processing.[4]

At the moment, we have implemented several sentence selection methods which are described in Mollá et al. (2007). To reduce variables in the experiments reported in this paper, we have decided to use the simplest selection method only, which is based on word overlap. Sentences are scored based on the number of words that can be found in both the question and the sentence. Stopwords, words occurring frequently, are not used in this score. Other methods implemented, but not used in the experiments, use richer linguistic information.

The sentences remaining after the sentence selection phase are then handed to a NER. All NEs found in these sentences are considered to be possible answers to the user question. Since we want to measure the impact of the NER in the QA system, we only use the NEs as possible answers. All other mechanisms of finding answers that are otherwise used in AnswerFinder are not used in these experiments.

In these experiments we do not use NEs to filter out sentences (the first case in section 2). Currently we are investigating possible ways to use the NE information for sentence selection.

Once all possible answers to the questions are found, the actual answer selection phase takes place. For this, the question is analysed, which provides information on what kind of answer is expected. Using this information, the possible answers that match the expected answer type are selected and scored.

Finally, the best answer is returned to the user. Best answer in this context is considered as the an-

[4]Selecting sentences out of context assumes that the answer to a question can be found in one sentence. At the moment, the system does not handle answers that are distributed over multiple sentences. Similarly, the system expects that the answer can be found in one document. This does not allow multi-document summaries as requested in the DUC summarisation competitions (Dang, 2006).

Class	Type	# in BBN
ENAMEX	Organization	30,248
	Person	13,751
	Location	14,656
TIMEX	Date	20,672
	Time	1,069
NUMEX	Money	11,097
	Percent	5,976

Table 1: Entities used in the MUC tasks and number of occurrences in the BBN corpus

swer with both the highest score and matching the answer type or simply the answer with the highest score if none of the possible answers fits the answer type.

4 Named Entity Recognition

We have tried different NERs in the context of question answering. In addition to a general purpose NER, we have developed our own NER. Even though several high quality NERs are available, we thought it important to have full control over the NER to make it better suited for the task at hand.

4.1 ANNIE

ANNIE is part of the Sheffield GATE (General Architecture for Text Engineering) system (Gaizauskas et al., 1996) and stands for "A Nearly-New IE system". This architecture does much more than we need, but it is possible to only extract the NER part of it. Unfortunately, there is not much documentation on the NER in ANNIE, so we will need to use it as a black box. The NE types found by ANNIE match up with the MUC types as described in Table 1.

ANNIE was chosen as an example of a typical NER because it is freely available to the research community and the NE types match with the MUC types.

4.2 AFNER

In addition to the ANNIE NER, we also looked at the results from the NER that is developed within the AnswerFinder project, called *AFNER*, which is based on machine learning. The technique used is maximum entropy, and the implementation of the classifier is adapted from Franz Josef Och's *YASMET*[5]. The system is trained on the BBN Pronoun Coreference and Entity Type Corpus (the BBN corpus in short), which is available at the Linguistics Data Consortium[6]. The NE hierarchy of the BBN corpus uses a much finer gradation than that of ANNIE. To enable a comparison between AFNER and ANNIE we have mapped the BBN corpus entities to the MUC set. Table 1 contains the number of entities in the BBN corpus after they have been mapped to the MUC set.

4.2.1 Features

The features used by AFNER combine regular expressions and gazetteers with properties internal and external to the token.

Regular expressions are useful for identifying specific patterns characteristic of some entity types such as dates, times, percentages, and monetary expressions. The range of entities that can be discovered using regular expressions is limited, but the precision of the regular expressions is high. Therefore, matching a particular regular expression is a key feature used in identifying entities of these particular types.

Gazetteers are useful for finding commonly referenced names. If an expression is found in one of the lists, then it is likely that the expression is of the type indicated by the list, but sometimes this is not the case. By introducing gazetteers as additional features in the classifier it becomes possible to use other features that may be more determinant for the categorisation of a specific token in particular cases. AFNER uses three lists (locations, person names, and organisations), with a total of about 55,000 list items.

Features relating internal token properties include those such as capitalisation, alpha/numeric information, etc. and are listed in Table 2.

In addition AFNER incorporates contextual features. These are features that identify a token amongst surrounding text, or relate a token to tokens in surrounding text. In particular AFNER applies a set of regular expressions to the neighbouring tokens within a context window and records the match results as features. These regular expressions detect patterns such as whether the neighbouring token is made of two digits, or whether it is a currency name. Additional features include the class assigned to the previous token and all of its class probabilities.

A final set of features relates to global informa-

[5] `http://www.fjoch.com/YASMET.html`

[6] `http://www.ldc.upenn.edu`

Regular Expressions	Specific patterns for dates, times, etc
FoundInList	The token is a member of a gazetteer
InitCaps	The first letter is a capital letter
AllCaps	The entire word is capitalised
MixedCaps	The word contains upper case and lower case letters
IsSentEnd	The token is an end of sentence character
InitCapPeriod	Starts with capital letter and ends with period
OneCap	The word is a single capitalised letter
ContainDigit	The word contains a digit
NumberString	The word is a number word (‘one’, ‘thousand’, etc.)
PrepPreceded	The word is preceded by a preposition (in a window of 4 tokens)
PrevClass	The class assigned to the previous token
ProbClass	The probability assigned to a particular class in the previous token
AlwaysCapped	The token is capitalised every time it appears

Table 2: Features used in AFNER

tion inspired on those features described by Chieu and Ng (2002). Currently AFNER only checks whether a token is always capitalised in a passage of text.

4.2.2 General Method

The features are passed to a maximum entropy classifier which, for each token, returns a list of probabilities of the token to pertain to each category. The categories correspond with each type of entity type prepended with ‘B’ and ‘I’ (indicating whether the token is at the beginning or inside a NE respectively), and a general ‘OUT’ category for tokens not in any entity. The list of entity types used is the same as in the MUC tasks (see Table 1).

The classifier returns an ordered list of tags for each token. We only allow tags that have a probability of more than half relative to the probability of the next highest tag. Additionally, we can set a maximum number of tags that are assigned to tokens. Preliminary experiments revealed that often the top two or three entity type probabilities have similar values.

By allowing tokens to have multiple tags assigned to it, AFNER aims at high recall. The presence of multiple tags means that NEs can be nested (AFNER may output NEs that are contained in other NEs). In the rest of the article, we allow AFNER to classify two tags at most for each token. Preliminary experiments have shown that this greatly increases recall, whereas using three tags for each token only increases recall a bit more while decreasing precision massively.

Classified tokens are then combined according to their classification to produce the final list of NEs. The general method is as follows. Each label prepended with ‘B’ signals the beginning of a possible NE of the relevant type and each ‘I’ label continues a NE if it is preceded by a ‘B’ or ‘I’ label of the same type. If an ‘I’ label does not appear after a ‘B’ classification, it is treated as a ‘B’ label. In addition, if a ‘B’ label is preceded by an ‘I’ label, it will be both added as a separate entity (with the previous entity ending) and appended to the previous entity. The result is a set of tags that may overlap (Figure 1). This is filtered by selecting the longest-spanning entity and discarding all substring or overlapping strings. If there are two entities associated with exactly the same string, the one with higher probability is chosen (Figure 2).

The probability of a multi-token entity is computed by combining the individual token probabilities. Currently we use the geometric mean but we are exploring other possibilities. If P_i is the probability of token i and $P_{1...n}$ is the probability of the entire sentence, the geometric mean of the probabilities is computed as:

$$P_{1...n} = e^{\frac{\sum_{i=1}^{n} \log P_i}{n}}$$

5 Results

To measure the impact of the quality of NERs and the quantity of the NEs generated in the context of question answering, we incorporated the two NERs in a question answering system. We first explain the experimental setup, followed by empirical results and a discussion.

BPER	ILOC						
IPER	BLOC			BLOC		BDATE	
BLOC	IPER	OUT	OUT	IPER	OUT	IDATE	OUT
Jack	*London*	*lived*	*in*	*Oakland*	*in*	*1885*	*.*
PERSON	LOCATION			LOCATION		DATE	
PERSON				PERSON			
LOCATION							

Figure 1: Named entities as multiple labels. The token-based labels appear above the words. The final NE labels appear below the words.

BPER	ILOC						
IPER	BLOC			BLOC		BDATE	
BLOC	IPER	OUT	OUT	IPER	OUT	IDATE	OUT
Jack	*London*	*lived*	*in*	*Oakland*	*in*	*1885*	*.*
PERSON				LOCATION		DATE	

Figure 2: Named entities as single labels. The token-based labels appear above the words. The resulting NE labels appear below the words.

5.1 Experimental setup

In this evaluation, we use the data provided for the question answering track of the 2005 TREC competition-based conference[7]. The data is provided specifically to measure performance of question answering systems and as such it provides a good benchmark set.

The document collection consists of just over one million news paper articles. Additionally, a list of 530 questions, grouped by topic, is provided. Each question can be one of three types: factoid (requiring a single fact as an answer), list (which may have multiple answers) and other (asking for any additional information on a topic). Here, we only consider the factoid questions, of which there are 362. To determine whether a question is answered correctly, we use Ken Litkowsky's answer patterns, which are also available from the TREC website.[8] For each topic, a list of relevant documents was generated by running the PRISE information retrieval engine on the question targets. This list of documents is also available from the TREC website. For each target, 1,000 documents were provided in the same order as returned by PRISE.

The question answering system used in the experiments is a simplification of AnswerFinder and works as follows. Firstly, the system selects documents that are relevant to a particular question using the list of relevant documents. From these documents, the best sentences are selected based on the size of the word overlap between the sentence and the question, where larger overlaps are better. The selected sentences are then provided as input for the different NERs, ANNIE and AFNER. This results in a list of NEs that the system takes as possible answers. Finally, the system selects only those NEs that match with the type of question and the NE with the highest score (which is the number of times the NE was found) is returned. If multiple NEs have the same score, one is selected at random.

The impact of each of the phases is computed by considering the percentage of questions that can still be answered after that phase. Additionally, we will provide information on the amount of text that is left with respect to the previous phase (as each phase removes unnecessary text).

5.2 Empirical results

Table 3 contains the results after document selection. The documents are selected according to the output of PRISE as described above. In particular we used the list of preselected documents provided with the TREC 2005 questions. The results presented here are computed by taking the top n documents from the list (where n is 10, 20, etc.).

The percentages indicate how many of all ques-

[7] `http://trec.nist.gov`

[8] We assume a question is answered correctly if the answer can be identified as a string in the document collection. We do not check whether the information in the context of the answer actually supports the string to be the correct answer.

# of documents	% of questions
10	75.5%
20	81.6%
30	86.9%
40	89.5%
50	92.1%
100	93.6%
500	95.6%
1,000	95.9%

Table 3: Percentage of factoid questions that can still be answered after document selection

# of sentences	% of questions	% of remaining text
10	49.9%	(2.4%)
20	62.0%	(3.7%)
30	65.4%	(4.9%)
40	68.8%	(6.1%)
50	70.8%	(7.1%)
60	73.0%	(8.1%)
70	73.7%	(8.9%)
80	75.0%	(9.8%)
90	76.0%	(10.6%)
100	76.2%	(11.4%)

Table 4: Percentage of factoid questions that can still be answered after selecting sentences from the top 50 documents and percentage of remaining text in brackets

tions can still be answered, i.e. in how many of the questions the answer can still be found in the text that is left. Obviously, increasing the number of retained documents increases the percentage. This percentage is an upper-bound for the rest of the phases. Note that even when all 1,000 documents selected by PRISE are used, not all questions can be answered.

We will continue with the top 50 documents after document selection. Considering all sentences in these documents, we only select the best *n* according to the word overlap metric. This metric counts the number of words that can be found in both question and sentence, not counting function words. The results after sentence selection can be found in Table 4. The percentages in brackets show the percentage of text that is retained with respect to the full amount of text of the top 50 documents.

It is interesting to see that there is a drop from

# of sent.	% of questions			
	ANNIE		AFNER	
10	34.4%	(14.4%)	37.0%	(22.5%)
20	42.8%	(14.9%)	45.9%	(23.0%)
30	45.9%	(15.0%)	50.0%	(23.1%)
40	47.8%	(14.9%)	52.3%	(23.0%)
50	49.3%	(15.0%)	53.7%	(23.1%)
60	50.8%	(15.0%)	55.1%	(23.1%)
70	52.0%	(15.0%)	55.9%	(23.1%)
80	52.6%	(15.1%)	57.0%	(23.2%)
90	52.8%	(15.2%)	58.2%	(23.3%)
100	52.8%	(15.2%)	58.2%	(23.4%)

Table 5: Percentage of factoid questions that can still be answered after NE recognition of the selected sentences from the top 50 documents and the percentage of text that is retained (in brackets)

92.1% answerable questions to 76.2% when selecting the best 100 sentences. However, this loss of answers can be expected, since we are only left with only 11.4% of the initial text. The relatively large drop in results can also be explained from the fact that the word overlap metric is crude. The scores given to the sentences are very coarse-grained and this phase can probably be improved using more sophisticated metrics. However, these figures serve as upper-bounds in the experiment.

The selected sentences are now handed to the two NERs. These find a list of NEs, which are taken as possible answers. Table 5 illustrates the percentage of questions that can still be answered together with the percentage of text remaining with respect to the amount of text in the selected sentences.

It has to be taken into account that there is an upper bound of questions that can be answered correctly that is given in Table 4. However, not all questions can be answered using NEs (especially taking only MUC NE types into account), so the actual upper bound is a bit lower.

The percentages of answerable questions given in Table 5 approximate recall. They indicate how many correct answers are identified by the NERs, whereas the percentages in brackets (again these are percentages of text retained with respect to the selected sentences) give a rough indication of (inverted) precision in that small percentages would ideally contain the correct entities only, whereas larger percentages contain more noise. The aim is therefore to reduce the text as much as possible

while retaining as many answerable questions as possible.

Table 5 shows that AFNER generates the highest number of potential answers (23.4% of the text is retained when using 100 sentences compared to ANNIE with 15.2%) and it also covers the highest number of questions (58.2% versus 52.8%). This means that AFNER has a higher recall (more questions can be answered), but it has a lower precision (more text is retained), whereas ANNIE has a lower recall, but retains less text.

The results provided in Table 5 illustrate that it is important to have a high recall. In a full-blown QA system, NEs found by ANNIE can be used to answer fewer questions that those found by AFNER. On the other hand, it may be the case that the additional text AFNER introduces works as noise that degrades the final results of the question answering system.

To measure the impact of additional text versus percentage of answerable questions, we test the different NERs in our simplified version of AnswerFinder. The NEs found as described above are filtered by removing those NEs that are not of the expected answer type. From the remaining NEs, the one that occurs most often in the text is selected. If there are several with the same score (say *n*), we count the number of those answers that are correct and divide them by *n*. The results of this system are shown in Table 6.

# of sentences	% of questions	
	ANNIE	AFNER
10	8.3%	6.0%
20	8.4%	6.7%
30	9.3%	6.5%
40	8.6%	7.3%
50	8.1%	7.0%
60	7.6%	7.3%
70	7.5%	6.8%
80	7.9%	6.4%
90	9.0%	6.5%
100	8.5%	6.1%

Table 6: Percentage of factoid questions that found an answer in a baseline question answering system given the top 50 documents

With respect to the results in Table 5, the figures in Table 6 are drastically lower. This illustrates that the final answer selection method is very crude. More sophisticated answer selection should improve on these results.

There does not seem to be a consistent line in each of the system's results. This indicates that both systems are very sensitive to the actual counts of the NEs they find in the sentences. Using more data clearly does not mean that better answers are found.

The results in Table 6 show that the advantage AFNER has in finding more answers to questions is reduced when the output is used directly in a QA system. The amount of noise introduced by increasing the recall has a negative effect on the system output.

6 Summary and Conclusion

In this paper we have introduced a NER developed within the context of QA. In our experiments we have tested the impact of the NERs available in ANNIE and AFNER, our custom-built system. The experiments show that AFNER's recall is higher than ANNIE's. The results also showed that there is a complex interaction between the effects of recall and performance when using NEs in a QA context. Whereas recall is important to be able to answer as many questions as possible, the amount of noise introduced to do this should be kept at a minimum.

A more detailed analysis of the impact of NEs quality in a QA system is needed. The means of selecting answers in the system described here is very crude and is highly dependable on the quality of the NEs to start with. Additional information that can be extracted during the QA system run should be able to identify correct answers better. In particular, we plan to test various versions of the complete AnswerFinder system (not just the baseline setting) with both NERs.

Even though the recall of AFNER is higher than that of ANNIE, the precision is probably too low. In Table 5 AFNER generated almost twice the amount of output of ANNIE. Part of this is noise that resulted in reduced scores in the final system. We are currently studying methods to increase the precision of AFNER while retaining the high recall, in order to build a NER that is better suited to QA.

Acknowledgements

This work is supported by the Australian Research Council under the ARC Discovery grant DP0450750.

Special acknowledgements are due to Daniel Smith and Alex Chilvers, who were involved in the development of AFNER.

References

Quintin Armour, Nathalie Japkowicz, and Stan Matwin. 2005. The role of named entities in text classification. In *Proceedings CLiNE 2005*. Gatineau, Canada. URL http://www.crtl.ca/cline05/cline05_papers/ArmourJapkowiczMatwin.pdf.

Haoi Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: A maximum entropy approach using global information. In *Proceedings COLING 2002*.

Hoa Tran Dang. 2006. DUC 2005: Evaluation of question-focused summarization systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, pages 48–55. Association for Computational Linguistics, Sydney. URL http://www.aclweb.org/anthology/W/W06/W06-0707.

Robert Gaizauskas, Hamish Cunningham, Yorick Wilks, Peter Rodgers, and Kevin Humphreys. 1996. GATE: an environment to support research and development in natural language engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence*. Toulouse, France. URL http://www.dcs.shef.ac.uk/research/groups/nlp/gate/.

Kevin Humphreys, George Demetriou, and Robert Gaizauskas. 2000. Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structures. In *Proceedings of the Pacific Symposium on Biocomputing'00 (PSB'00)*, pages 502–513. Honolulu, Hawaii. URL http://www.bionlp.org/psb2000/humphreys.pdf.

Diego Mollá and Menno van Zaanen. 2005. Learning of graph rules for question answering. In Tim Baldwin and Menno van Zaanen, editors, *Proc. ALTW 2005*. ALTA. URL http://www.alta.asn.au/.

Diego Mollá, Menno van Zaanen, and Luiz Pizzato. 2007. Answerfinder at TREC 2006. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings TREC 2006*. 8 pages.

Elisa Noguera, Antonio Toral, Fernando Llopis, and Rafael Muñoz. 2005. Reducing question answering input data using named entity recognition. In *Proceedings of the 8th International Conference on Text, Speech & Dialogue*, pages 428–434. URL http://dx.doi.org/10.1007/11551874_55.

Beth M. Sundheim. 1995. Overview of results of the MUC-6 evaluation. In *Proc. Sixth Message Understanding Conference MUC-6*. Morgan Kaufmann Publishers, Inc. URL http://acl.ldc.upenn.edu/M/M95/M95-1002.pdf.

Ellen M. Voorhees. 1999. The TREC-8 question answering track report. In Ellen M. Voorhees and Donna K. Harman, editors, *Proc. TREC-8*, number 500-246 in NIST Special Publication. NIST. URL http://trec.nist.gov/pubs.html.

Effect of combining different Web search engines on Web question-answering

Tatsunori Mori Akira Kanai Madoka Ishioroshi Mitsuru Sato
Graduate School of Environment and Information Sciences
Yokohama National University
79-7 Tokiwadai, Hodogaya-ku, Yokohama 240-8501, Japan
{mori,a-kanai,ishioroshi,mitsuru}@forest.eis.ynu.ac.jp

Abstract

We have several different commercial Web search engines that are available. It is expected that the combination of the results from different Web search engines has some impact on the accuracy of question-answering (QA) for Web documents, because the search results are not identical and the combination increases the variety of information source. However, as far as we know, there are no studies on the effect of combining different Web search engines on QA.

In this paper, we examined the effect of the combination on QA. We investigated three different methods to combine search results from different search engines in the process of QA. The first one is a conventional method that straightforwardly merges search results from different search engines, then, feeds the unified search result into one QA engine. On the other hand, the second one and third one are our proposal methods that feed each search result from individual search engine to a QA engine separately, then merge the answer candidates. The experimental result showed that the methods that merge the answer candidates after QA are more effective than the method that merges the search results before QA.

1 Introduction

The technology of question-answering (QA) is widely regarded as an advancement on the combination of information retrieval (IR) and information extraction (IE). QA systems do not provide us with the relevant documents; instead, they directly provide answers to questions. For example, when the system receives the question "What is the capital of Japan?", it searches 'knowledge resource' for answer candidates, and hopefully, it would return the answer "Tokyo."

With regard to knowledge resource, while earlier QA systems utilized static document collection such as a set of newspaper articles, many of recent studies focus on Web documents because the Web is an up-to-date information resource. We term the question-answering with Web documents *Web QA* in this paper. Since it is not realistic that QA services prepare their own Web crawler and Web search engine, existing commercial search engines are borrowed for Web QA systems. Fortunately, many of major search engine companies publicly offer their own APIs (application interfaces) to users.

Here, we would like to pay attention to the fact that we have *multiple different* Web search engines that are available for Web QA. It is expected that the combination of the results from different Web search engines has some impact on the accuracy of Web QA, because the search results are not identical and the combination increases the variety of information source.

However, as far as we know, there are no studies on the effect of combining different Web search engines on QA. In this paper, we examined the effect of the combination on QA. We will investigate three different methods to combine search results from different search engines in the process of QA. The first one is a conventional method that straightforwardly merges search results from different search engines, then, feeds the unified search result into one QA engine. On the other hand, the second one and third one are methods that feed each search result from individual search engine to a QA engine separately, then merge the answer candidates.

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 325–332,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

2 Related work

Lin et al.(Lin and Katz, 2003) give us a good tutorial about Web QA. According to it, there are at least two ways of using Web documents. First one is to use Web documents as the primary corpus of information. Second one is to combine use of Web documents with other corpora. In this paper, we focus on the first way of use.

It is pointed out that one advantage of use of Web document is the data redundancy of Web documents. The expressiveness of natural language allows us to say the same thing in multiple ways. The fact is usually one main problem of question answering, because an answer may be stated in different ways from a question. However, with data redundancy on the Web, it is likely that the answer will be stated in the same way as the question was asked(Lin and Katz, 2003), because a lot of different authors may describe the answer in their own ways. That is, the data redundancy gives us the variety of description.

From the viewpoint of variety of description, several researches take advantage of variety of information source. For example, recent version of START, which is one of the first Web-based QA system, makes use of multiple information source(Katz et al., 2004, 2005). Radev et al. proposed a probabilistic approach to question answering on the Web(Radev et al., 2005), and they uses three major search engines to retrieve the top 40 documents.

Although these researches utilize documents from different information sources, they make no distinction between information sources after document retrieval. On the other hand, our methods described in Section 4 try to exploit the data redundancy among the difference information sources.

3 Basic question-answering system

The basic QA system used in this study is a real-time QA system based on the study by Mori (Mori, 2005). It can answer Japanese factoid questions in Japanese. As shown in Figure 1, the system comprises six parts of process — the question analysis, interface to external search engine, passage extraction, sentential matching, answer generation, and pseudo voting.

The process of question analysis receives a question from a user and extracts several kinds of information including a list of keywords, the question type, and so on. Here, we define the term *Keywords* as content words in a given question. The list of keywords is submitted to an external search engine to retrieve relevant documents.

The process of sentential matching receives a set of sentences from the passage extractor. It treats each morpheme as an answer candidate and assigns it a matching score as described below. It should be noted that a morpheme may be either a word or a part of a longer compound word. Therefore, in the latter case, the process of answer generation extracts the compound word including the answer candidate, and treats it as a proper answer candidate.

3.1 Raw scores for answer candidates

In the basic QA system, a composite matching score for an answer candidate is adopted as shown in Equation (1). We term it *raw score* in this paper. It is a linear combination of the following sub-scores for an answer candidate AC in the i-th retrieved sentence L_i with respect to a question sentence L_q:

1. $Sb(AC, L_i, L_q)$, the matching score in terms of character 2-grams;

2. $Sk(AC, L_i, L_q)$, the matching score in terms of the keywords;

3. $Sd(AC, L_i, L_q)$, the matching score in terms of the dependency relations between an answer candidate and the keywords; and

4. $St(AC, L_i, L_q)$, the matching score in terms of the question type.

In the calculation of $St(AC, L_i, L_q)$, we employ an NE recognizer that identifies eight types of NEs defined in the IREX-NE task(IREX Committee, 1999).

$$\begin{aligned} S(AC, L_i, L_q) = & \\ & Sb(AC, L_i, L_q) + Sk(AC, L_i, L_q) + \\ & Sd(AC, L_i, L_q) + St(AC, L_i, L_q) \end{aligned} \quad (1)$$

In order to reduce the computational cost, the A* search control is introduced in the sentential matching mechanism. With this control, the system can process the most promising candidate first, while delaying the processing of the other candidates, and perform the n-best search for the answer candidates.

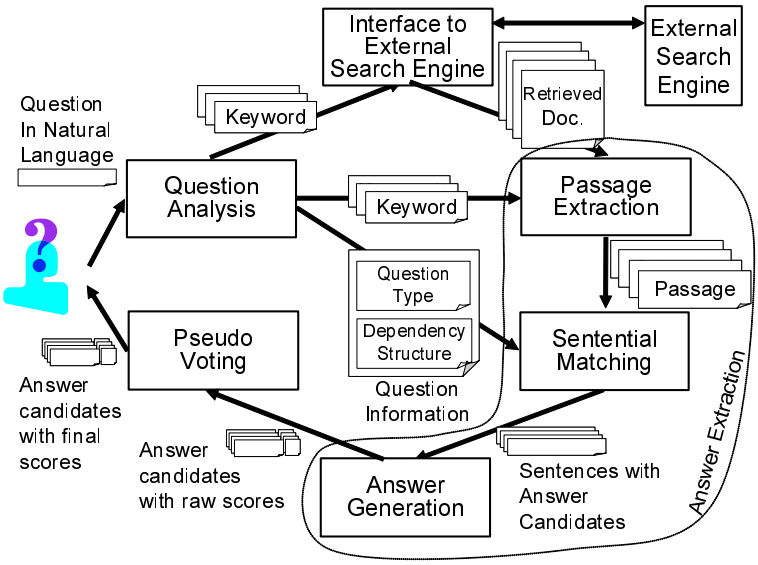


Figure 1: Basic QA system

3.2 Pseudo voting method in search scheme

Many existing QA systems exploit global information on answer candidates. In particular, redundancy is the most basic and important information. For example, there are previous studies that boost the score for answer candidates that occur multiple times in documents(Clarke et al., 2001; Xu et al., 2003). This is known as the *voting method*.

On the other hand, we cannot exploit the voting method directly in the scheme of searching answers because the system quits the searching after n-best answers are found. Therefore, an approximation of the voting method, termed *pseudo voting*, is introduced as follows. In case n-best answers are necessary, the system continues searching for answers until n *different* answer candidates are found. Therefore, the system may find other answer candidates that have the same surface expression as one of the answer candidates that have already reached the goal state of search. Consequently, we can partially use the frequency information of answer candidates by recording all that have reached the goal state in the search process. In this paper, the pseudo voting score $S^v(AC, L_q)$ for an answer candidate AC is defined as follows:

$$S^v(AC, L_q) = (\log_{10}(freq(AC, AnsList)) + 1) \cdot \max_{L_i} S(AC, L_i, L_q) \quad (2)$$

where $AnsList$ is the list of answer candidates that have reached the goal state in the n-best search, and $freq(x, L)$ is the frequency of x in L. We also term the pseudo voting score *the final score* in this paper.

According to the experiments by Murata et al.(Murata et al., 2005), the above voting score is comparable with other good voting scores.

3.3 Exploiting Web documents by using snippets in Web search results

By replacing the document search engine with a Web search engine, the basic QA system can be easily exploit Web documents. However, downloading a couple of hundred Web documents is time-consuming task. To address the problem, Sagara et al.(Sagara et al., 2006) empirically showed that the use of snippets, which are short extractive summary produced by Web search engine, are effective as the resource for Web QA. Katz et al.(Katz et al., 2005) also use the snippets to generate answer candidates. We follow the same line in order to reduce the turn-around time.

Figure 2 shows the structure of the basic Web QA system, which utilizes the snippets from a search engine and has the basic QA system described above as a QA engine. In this figure, "the wrapper program for Web Search Engine" is a program component that absorbs the difference of protocols between each API of a particular search engine and the basic QA engine. It first receives a query from the basic QA engine, and then submits the list of keywords as the query to a particular search engine. Second, it extracts snippets from the search result and returns the snippets to the ba-

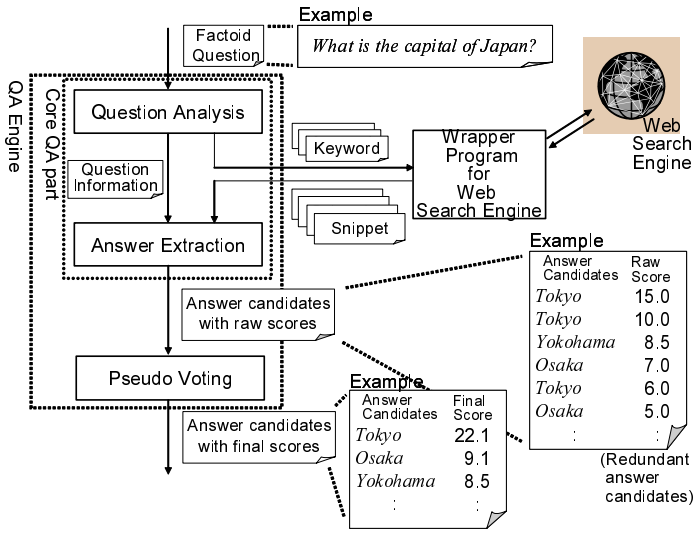


Figure 2: Basic Web QA system

sic QA engine as a set of documents. Finally, the basic QA engine performs question-answering.

4 Methods to combine search results from different Web search engines

There are, at least, the following three methods to combine the results from different Web search engines.

Combination A: the method that straightforwardly merges search results from different search engines, then, feeds the unified search result into one QA engine.

Combination B: the method that feeds each search result from individual search engine to a QA engine separately, and then merges the answer candidates with raw scores before pseudo voting.

Combination C: the method that feeds each search result from individual search engine to a QA engine separately, and then merges the answer candidates after pseudo voting. Since here we may have redundant answer candidate, the second voting is performed with Equation (2). Here, 'the raw score' for the second voting is the final score after the first pseudo voting.

Figures 3, 4 and 5 show the combination A, B, and C, respectively. In these figures, a "merger" receives lists of data from multiple sources and just merges the lists.

The combination A is the baseline method, which merges the result from different search engines *before* the question-answering process. The same kind of approaches is adopted in other researches as described in Section 2. The combinations B and C are our proposed methods, which merge the results from different search engines not before the question-answering but *after* the core question-answering process.

It should be noted that Combination B and C are different from Combination A from the viewpoint of the variety of information source. Since both Combination A and Combination B extract answer candidates with higher raw scores from snippets produced by Web search engines, they may be supposed to produce the almost same answer candidates. However, in Combination A, all answer candidates are treated equally in the ranking regardless of the difference of information source. Therefore, the answer candidates that are forwarded to the pseudo voting process may be resulted only from a small number of search engines. On the other hand, in Combination B, the answer candidates to be voted come evenly from all search engines. Thus, the variety of information source is ensured. Combination C is also in the same situation as Combination B. Moreover, in Combination C, answer candidates from many different information source would receive higher score in the stage of the second voting.

Here, we have a hypothesis that the certainty of answer candidates can be measured by the variety of information sources. If the hypothesis holds, we

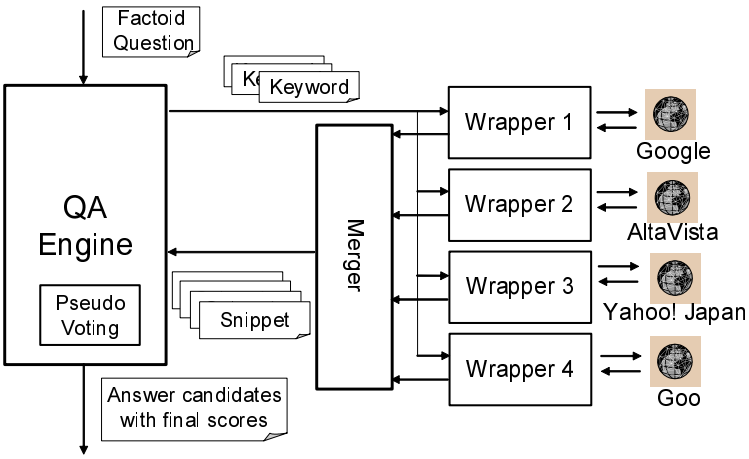


Figure 3: Baseline QA system with multiple search engines (Combination A)

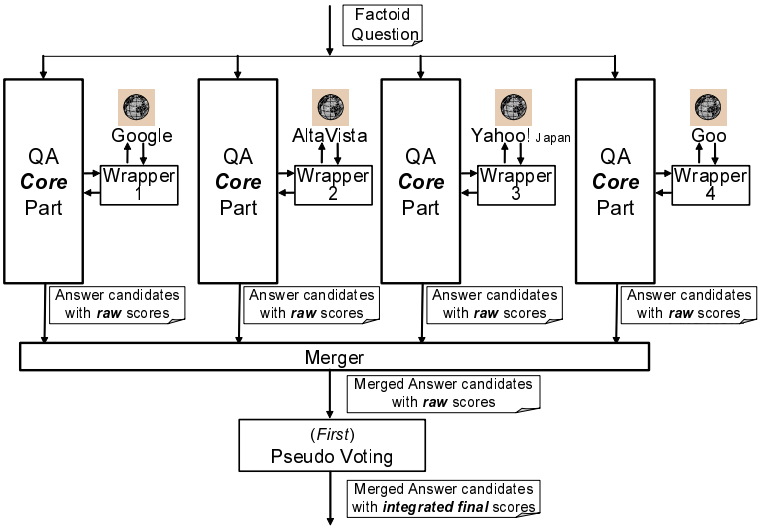


Figure 4: QA system with multiple search engines (Combination B)

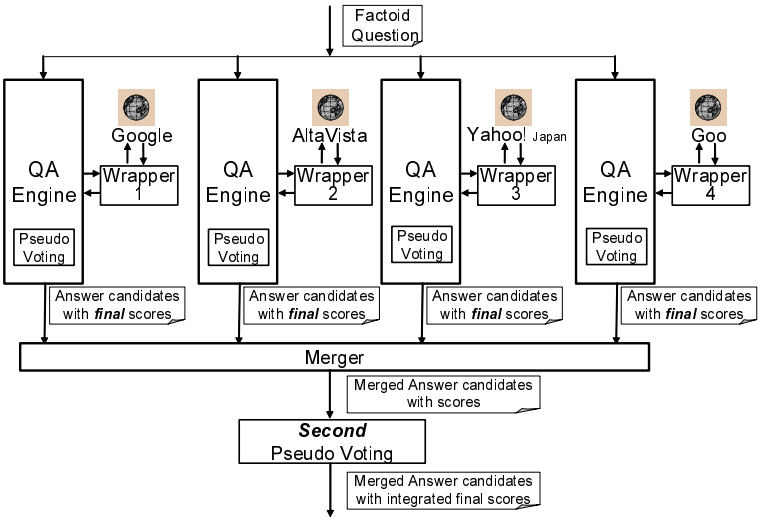


Figure 5: QA system with multiple search engines (Combination C)

may expect that Combination B and C are more accurate than Combination A.

5 Experimental result

In order to evaluate three combination methods, we conducted question-answering experiments as described below. In the same condition, we also examine the accuracy of the systems that use only one of the search engines as baselines.

5.1 Question set and their answer set

As for the question set and their answer set, we use a subset of the question set of NTCIR-3 QAC1(Fukumoto et al., 2002) and their corresponding official answer strings. NTCIR QAC is a series of evaluation workshops for question-answering organized by the National Institute of Informatics, Japan. The question set of NTCIR-3 QAC1, which consists of factoid-type questions, is originally designed for evaluating Japanese QA systems that have Japanese newspaper articles in 1998 and 1999. Therefore, the *current* answers of some questions have been changed from the official answers[1]. However, we would not like to judge the answer candidate with our own criterion, because of the neutrality and objectivity of the evaluation. Therefore, we just skipped this sort of questions.

The other issue is the problem of hanging-up of APIs. It is very rare, but some APIs of Web search engines we used hung up by the merest chance. In this evaluation, when at least one API timed out, we also skip the question.

By skipping these sort of problematic questions, from the question set of QAC1, we selected 50 questions that have earlier ID numbers. The actual question set is shown in the appendix as a list of question IDs.

5.2 Other experimental settings

With regard to Web search engines, we made use of the following Japanese-capable Web search engines as external search engines:

1. Google (`http://www.google.co.jp/`, Google SOAP Search API),
2. goo (`http://www.goo.ne.jp/`),
3. AltaVista (`http://www.altavista.com/`),
4. Yahoo! Japan (`http://www.yahoo.co.jp/`, Yahoo! JAPAN Developer Network).

All search results were obtained in February 24th and 25th, 2007.

As for the parameters related to the QA engine, the setting shown in Table 1 is used. According to Mori (Mori, 2005), the combination of the values is effective, at least, when the information source is a set of newspaper articles.

Here, it should be noted that the total number of documents for the systems of the combination methods is four times larger than the systems with only one of search engines, because the combination methods are utilize four different search engines with the same value of the parameter d. Although it is a very difficult problem to tell what is the fair comparison, one other possible choice of setting would be that the number of documents for the methods with only one search engine is increased four times. However, Mori (Mori, 2005) reported that increasing the number of documents does not necessarily improve the accuracy of question-answering because the lower ranked documents may eventually yield incorrect answer candidates that have lower raw scores but have high frequencies. Finally, we straightforwardly adopted the original setting of the parameters in Table 1.

5.3 Results

The accuracy of each combination method was evaluated using the mean reciprocal rank (MRR) and the average accuracy of the top n answer candidates. Reciprocal rank (RR) is the inverse of the rank of the first correct answer for each question. If no correct answer appears within the top five answer candidates, RR is 0. MRR is the average of RR over all questions. On the other hand, "Top n" represents the ratio of the number of questions whose one correct answer is found within the top n answer candidates.

The evaluation results are summarized in Tables 2 and 3.

6 Discussion

As shown in Table 2, the combination methods outperform methods that utilize only one Web search engine. Combination B and C in particular have good performance in terms of accuracy.

[1] For example, the answer to the question "What is the population of Tunisia?" (the question ID is QAC1-1019-01 and it is originally written in Japanese.) should vary year by year when we use Web documents as information source.

Table 1: Parameter setting of the QA engine for experiments

Parameter name	Value	Description
a	10	Number of answers to be searched.
d	250	Number of documents (snippets) to be retrieved.
ppd	5	Maximum number of passages retrieved from one document (snippet).
p	30	Number of passages to be considered in the retrieved documents (snippets).

Table 2: Accuracy of each combination methods

Method	MRR	Accuracy				
		Top 1	Top 2	Top 3	Top 4	Top 5
Google only	0.349	0.280 (14/50)	0.360 (18/50)	0.420 (21/50)	0.440 (22/50)	0.460 (23/50)
Goo only	0.314	0.220 (11/50)	0.300 (15/50)	0.420 (21/50)	0.260 (23/50)	0.480 (24/50)
AltaVista only	0.356	0.300 (15/50)	0.380 (19/50)	0.400 (20/50)	0.420 (21/50)	0.440 (22/50)
Yahoo! Japan only	0.376	0.300 (15/50)	0.420 (21/50)	0.440 (22/50)	0.460 (23/50)	0.480 (24/50)
Combination A	0.392	0.320 (16/50)	0.400 (20/50)	0.480 (24/50)	0.500 (25/50)	0.500 (25/50)
Combination B	**0.456**	**0.360** (18/50)	**0.480**(24/50)	**0.560** (28/50)	**0.580** (29/50)	**0.600** (30/50)
Combination C	0.451	**0.360** (18/50)	**0.480** (24/50)	0.520 (26/50)	0.560 (28/50)	**0.600** (30/50)

Although Combination B was well performed than Combination C, the difference is not so significant.

The result shows that the methods that merge the answer candidates after question answering is more effective than the method that merge the search results before question answering. As described in Section 4, in the former methods, i.e. Combination B and C, the answer candidates to be voted come evenly from all search engines, and the variety of information source is ensured.

According to Table 3, there are 33 questions for which at least one system in Table 2 finds correct answers within Top 5. While the systems with only one search engine found correct answers only for 22-to-24 questions within Top 5, Combination B and C improved the accuracy and could correctly answer 30 questions. From the fact, we may conclude that answer candidates from different search engines can complete each other. The fact also appears to support our hypothesis described in Section 4, namely, the hypothesis that the certainty of answer candidates can be measured by the variety of information sources.

7 Conclusion

We examined the effect of combining different Web search engines on QA. We investigated three different methods to combine search results from different search engines in the process of QA. The experimental result showed that the methods that merge the answer candidates after QA are more effective than the method that merges the search results before QA.

In this paper, we investigated only three primitive combination methods. The development of more effective combination method should be included in our future work.

Acknowledgements

We are grateful to the task organizers of NTCIR-3 QAC1 and people who manage the NTCIR workshops. We would like to especially thank Mainichi Shimbun and Yomiuri Shimbun for permitting us to use the documents for research. We would also like to thank the anonymous reviewers for their helpful comments.

This study was partially supported by Grant-in-Aid for Scientific Research on Priority Areas (No.18049031 and No.19024033) from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

Charles L.A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of SIGIR '01: the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 358–365.

Jun'ichi Fukumoto, Tsuneaki Kato, and Fumito Masui. 2002. Question Answering Challenge (QAC-1) — Question answering evaluation at NTCIR Workshop 3 —. In *Working Notes of the Third NTCIR Workshop meeting – Part IV: Question Answering Challenge (QAC1)*, pages 1–6.

IREX Committee, editor. 1999. *Proceedings of IREX workshop*. IREX Committee. (in Japanese).

Table 3: Distribution of questions in terms of difficulty in answering

Number of questions for which at least one system in Table 2 finds correct answers within Top 5	33/50
Number of questions for which all systems in Table 2 find correct answers at Top 1	7/50

B. Katz, M. Bilotti, S. Felshin, A. Fernandes, W. Hildebrandt, R. Katzir, J. Lin, D. Loreto, G. Marton, F. Mora, and O. Uzuner. 2004. Answering multiple questions on a topic from heterogeneous resources. In *Proceedings of TREC 2004*.

B. Katz, G. Marton, G. Borchardt, A. Brownell, S. Felshin, D. Loreto, J. Louis-Rosenberg, B. Lu, F. Mora, S. Stiller, O. Uzuner, and A. Wilcox. 2005. External knowledge sources for question answering. In *Proceedings of TREC 2005*.

Jimmy Lin and Boris Katz. 2003. Question answering techniques for the world wide web. URL `http://acl.ldc.upenn.edu/eacl2003/papers/tutorial/t2.pdf`, (Tutorial presentation at The 11th Conference of EACL (EACL-2003)).

Tatsunori Mori. 2005. Japanese question-answering system using A* search and its improvement. *ACM Transactions on Asian Language Information Processing (TALIP)*, 4(3):280–304. URL `http://portal.acm.org/TALIP/`.

Masaki Murata, Masao Utiyama, and Hitoshi Isahara. 2005. Use of multiple documents as evidence with decreased adding in a japanese question-answering system. *Journal of Natural Language Processing*, 12(2):209–247.

Dragomir R. Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. 2005. Probabilistic question answering on the web. *Journal of the American Society for Information Science and Technology*, 56(3).

Haruki Sagara, Tatsunori Mori, and Hiroshi Nakagawa. 2006. Effectiveness of snippets of web search engine as a knowledge source for qa. In *Proceedings of the twelfth annual meeting of the Association for Natural Language Processing*, pages 316–319. (in Japanese).

Jinxi Xu, Ana Licuanan, and Ralph Weischedel. 2003. TREC2003 QA at BBN: Answering definitional questions. In *Proceedings of the twelfth Text Retrieval Conferene (TREC 2003)*.

A Questions used in the experiment

We use the following 50 questions from NTCIR-3 QAC1. The full set of questions is available at the online proceedings of NTCIR Workshop 3 (Formal Run Questions of Task 1 in Question Answering Task, `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings3/`):

QAC1-1006-01, QAC1-1009-01, QAC1-1010-01, QAC1-1016-01, QAC1-1017-01, QAC1-1020-01, QAC1-1021-01, QAC1-1022-01, QAC1-1023-01, QAC1-1024-01, QAC1-1025-01, QAC1-1026-01, QAC1-1027-01, QAC1-1028-01, QAC1-1029-01, QAC1-1030-01, QAC1-1031-01, QAC1-1032-01, QAC1-1033-01, QAC1-1034-01, QAC1-1035-01, QAC1-1036-01, QAC1-1037-01, QAC1-1038-01, QAC1-1039-01, QAC1-1040-01, QAC1-1041-01, QAC1-1042-01, QAC1-1043-01, QAC1-1045-01, QAC1-1046-01, QAC1-1047-01, QAC1-1048-01, QAC1-1050-01, QAC1-1051-01, QAC1-1054-01, QAC1-1055-01, QAC1-1056-01, QAC1-1057-01, QAC1-1058-01, QAC1-1059-01, QAC1-1060-01, QAC1-1064-01, QAC1-1066-01, QAC1-1067-01, QAC1-1068-01, QAC1-1069-01, QAC1-1070-01, QAC1-1071-01, QAC1-1073-01.

What to Classify and How: Experiments in question classification for Japanese

Rebecca Dridan*
Computational Linguistics
Saarland University
rdrid@coli.uni-sb.de

Timothy Baldwin
CSSE Department
The University of Melbourne
tim@csse.unimelb.edu.au

Abstract

This paper describes experiments in Japanese question classification, comparing methods based on pattern matching and machine learning. Classification is attempted over named entity taxonomies of various sizes and shapes. Results show that the machine learning based method achieves much better accuracy than pattern matching, even with a relatively small amount of training data. Larger taxonomies lead to lower overall accuracy, but, interestingly, result in higher classification accuracy on key classes.

1 Introduction

Question classification is the primary aim of the question analysis stage of a question answering (QA) system. The goal of a QA system is to return an answer to a question framed in natural language and question classification aims to narrow down the possible types of answer that are likely, given the question. These answer types are commonly called *expected answer types* (EAT) and can range from classifications as broad as PERSON, NUMBER, LOCATION to those as specific as BASEBALL TEAM.

These answer types are generally referred to as named entity (NE) types, although they also include classes for dates and numbers rather than just entities referred to by name. A set of named entity types that is being used for question classification is known as a named entity taxonomy. The taxonomies used in various QA systems vary widely in size, and can be organised as a flat set or in a hierarchy. There are differences of opinion over the optimal size of a NE taxonomy used in question classification. Proponents of small (8–12) type taxonomies (e.g. Kurata et al. 2004) claim that large taxonomies lead to less accurate classification and hence less accurate QA results. Those who use larger sets (e.g. Breck et al. 2001) believe that the inaccuracies in classification are offset by the discriminative power of more types.

Methods for determining these EATs can be classified (as is common in many NLP tasks) as statistical (machine learning) or symbolic (most often pattern matching). Li and Roth (2006) showed that machine learning was a very effective method for question classification, using a test set of 1000 English questions. Despite these results, less than a quarter of the systems at the most recent QA evaluation forums (TREC 2006, CLEF 2006 and NTCIR 5) used machine learning to find EATs. One reason for this might be a perceived lack of training data. The experiment in Li and Roth (2006) used 21,500 English questions annotated with answer type; often this quantity of data is just not available, particularly for languages other than English. One goal then of the experiments reported here is to investigate whether machine learning is viable for question classification when much less data is available, in this case 2000 annotated Japanese questions. The second aim is to see how accuracy is affected by the size of the NE taxonomy.

2 The Data

2.1 Question Sets

The data set used for development is a set of 2000 Japanese questions described in Sekine et al. (2002b). Each question has been annotated with question type (WHO, WHAT, WHEN, ...) as well as the EAT taken from Sekine's Extended Named Entity (ENE) hierarchy (Sekine et al., 2002a). A second smaller set of 100 questions taken from the QAC-1 evaluation (Fukumoto et al., 2002) was manually annotated as a held out test set.

*This research was carried out while the first author was a student at The University of Melbourne.

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 333–341,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

	Size	Flat?	Comment
NE4	4	Yes	based on Li and Roth's coarse grained taxonomy
NE5	5	Yes	NE4 with ORGANIZATION added
IREX	9	Yes	standard IREX supplemented with a NUMERIC class
NE15	15	Yes	drawn from top two levels of ENE
NE28	28	No	augmented NE15, hierarchically splitting NUMBER, TIME and TIME PERIOD
ENE	148	No	version 4 of Sekine's ENE

Table 1: Summary of the characteristics of each Named Entity taxonomy used for question classification in these experiments

2.2 Named Entity Taxonomies

A set of six named entity taxonomies was defined; this was designed to investigate the effects of using taxonomies of different sizes and shapes. As the question data is annotated with types from Sekine's ENE hierarchy, all taxonomies are based on this hierarchy, and were created by merging various types into set categories. Two of the larger taxonomies are hierarchical; this allows us to investigate how the accuracy is affected if soft decision making is used later in the question answering process. Soft decision making in this instance means that if the predicted EAT does not match an answer candidate, then types around that EAT in the hierarchy may be considered, according to a defined strategy. We use a *most likely type* strategy in evaluation, where the most likely type is selected during classification, but direct ancestors or siblings of that type are also accepted. Other strategies might involve selecting the type that is the lowest common parent of likely types. The six taxonomies are described below, with their vital statistics summarised in Table 1.

The experiments in Li and Roth (2006) used both a coarse and a fine-grained taxonomy. In order to make some rough comparisons with this work, we have tried to approximate their coarse-grained set, which consisted of ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION and NUMERIC. None of the types in the ENE correspond to ABBREVIATION or DESCRIPTION, so we have defined two small sets, NE4 and NE5, where NE4 consists of THING, PERSON, LOCATION and NUMBER, and NE5 adds an ORGANIZATION class. In NE4, organisations are considered part of the PERSON class. Both of these sets are flat taxonomies.

The IREX named entity set (ORGANIZATION, PERSON, LOCATION, ARTIFACT, DATE, TIME, MONEY and PERCENT) was originally defined for an NE extraction task in the Japanese IREX project in 1998 (Sekine and Isahara, 1999). It has since been one of the standard NE type sets used in systems at the NTCIR workshops. We have defined a flat taxonomy of 9 types, based on the IREX set, but with an added NUMBER class for any numbers not related to money or percentages.

The taxonomies NE15 and NE28 are mid-range taxonomies that allow more expressive type labels, without being as fine-grained as the full ENE. NE15 is selected mostly from the top and second layer of ENE and is a flat taxonomy. NE28 expands on NE15 by allowing further divisions of NUMBER, TIME and TIME PERIOD, in a hierarchical manner. The sixth taxonomy used is the full ENE hierarchy, version 4, which is hierarchical and contains 148 types.

3 Pattern Matching

Rule based classifiers based on pattern matching were the most common question classification technique used at all the recent QA evaluations. To investigate both the issues involved and the accuracy achievable with this technique, the first question classification experiment used a set of manually created pattern matching rules for Japanese.

This task highlighted some of the issues that are peculiar to Japanese NLP. The first issue occurs because Japanese, like Chinese and Thai, is a non-segmenting language and hence questions need to be tokenised into 'words' before further processing. Various tokenisation tools exist for Japanese but there is no definition for the ideal lexical unit and so what constitutes a token varies between tools. Some Japanese processing systems split text into *bunsetsu*, which are logically larger than English words, similar to phrase chunks, while most of the standard tokenisers use smaller units, typically morphemes. Even within the morpheme based tools, tokenisation is inconsistent and it is necessary to use the same tokenisation tool for a whole system to avoid the systematic differences that occur when using different tokenisers. In all of the following experiments, we use ChaSen (Matsumoto et al., 1999) for tokenisation.

Other issues arise from the characters used to write Japanese. Japanese can be written using

4 different scripts—kanji, hiragana, katakana and the Latin alphabet. Kanji are Chinese characters which can have multiple pronunciations and are logographic in nature. There are thousands of kanji characters, although only 2000–3000 are commonly used. Hiragana and katakana are both phonetic scripts with 46 basic characters each. Hiragana is used for inflected verb endings, particles and other words where the kanji is either not known, or too formal. Katakana, on the other hand, is generally only used for foreign loanwords, or sometimes (e.g. in advertising) for emphasis. The Latin alphabet is often used in Japanese, particularly for acronyms. While there are kanji representations for numbers, the Arabic numbers are also often used.

All these scripts create two main problems – the first is that a single word can be written in different alphabets and so word based pattern matching rules have to accommodate this. The second problem is that there are too many characters to be represented by the ASCII character set. Different encoding systems exist that can represent Japanese, including UTF-8, which is slowly becoming an international standard and EUC-JP, which is the most common encoding system in Japan. While it is possible to convert from one encoding to another, this adds an extra layer of complexity and requires knowing in advance what encoding is used in a particular data set or tool. All of these issues affected the design of the pattern matching rules used, and so while the logic of the rules is very simple, the rules are complex and took many days to construct. The rules also ended up being very specific to the encoding system and the tokeniser used.

The basic logic of the rules first determines which one of 16 *question types* (e.g. 誰 *dare* "who", いくら *ikura* "how much", 何時 *nanji* "what time") applies to a question, by matching different variations of question words. Questions that contain 何 *nani* "what" are further processed so that fragments such as "what place" are classified as WHERE. Using this method question type can be determined with an accuracy of 96-99%. An abbreviated example of a rule for determining question type is shown in Algorithm 1.

Once the question type is determined, different rule sets are used depending on this type. Most of these rules looked for a word at a particular distance from the question word (called here the question focus), and checked whether this word was a recognised named entity type. Algorithm 2 shows one such rule. In addition, a small lookup table was hand-constructed that associated common synonyms for named entity types with their type. Some examples from the table are in Figure 1. It was difficult to construct such a set that improved accuracy but was not over-fitted to the data, and so the set remained small, containing only very specific terms.

Algorithm 1 Question type rule example

if *word*[*i*] starts with 何 or *word*[*i*] =なに or *word*[*i*] =なん or *word*[*i*] = なんと or *word*[*i*] = なんで **then**
  if *word*[*i*+1] = 処 or *word*[*i*+1] = 所 **then**
    return doko
  else if *word*[*i*+1] = 年ぶり or *word*[*i*+1] = 年度 or *word*[*i*+1] = 年 or *word*[*i*+1] = 日 or *word*[*i*+1] = 年前 or *word*[*i*+1] contains 月 or *word*[*i*+1] = ねん **then**
    return nannen
  else if *word*[*i*+1] = ドル or *word*[*i*+1] = 円 **then**
    return ikura
  else if ... **then**
    ...
  else
    return nani
  end if
end if

Algorithm 2 Answer type rule example

if *qtype* = *dono* **then**
  qfocus ← *qtype_index*+1
  if *word[qfocus]* in *lookuptable* **then**
    qfocus ← lookup(*qfocus*)
  end if
  if *qfocus* in *taxonomy* **then**
    return *qfocus*
  end if
  return country ▷ default for this type
end if

If the question focus could not be determined, the EAT was classified as the most likely type for that question type, given none of the previous rules applied. Results for this classification method are described in §5.

Canonical	Alternative
テレビ番組名	番組名
ポイント	点数
物質名	ミネラル
音楽名	曲

Figure 1: Lookup table examples: includes abbreviations, alternative orthography and synonyms.

4 Machine Learning

The learner used for the machine learning classification experiments was TiMBL version 5.1, a memory based learner (Daelemans et al., 2004). TiMBL classifiers can be trained very quickly, and hence allow a fast turnaround when evaluating the effects of different features. The learning algorithm used was TiMBL's IB1 algorithm, with $k = 1$ and features weighted by their Gain Ratio value. While some of the features (detailed later) were binary, many were symbolic, such as words or named entity types. In order to use the information that, for example, `country` is more similar to `province` than `person`, the modified value difference metric (MVDM) was used as a similarity measure between feature values.

The classifiers were evaluated first using a leave-one-out cross-validation strategy (train on 1999, test on one), for each of the 2000 questions in the development set. Next, to see how well the training generalised to unseen data, a separate evaluation was run on the 100 question held out set, using the 2000 questions as training data.

4.1 Features

The feature sets all had the same basic format: presence or absence of each of a list of question words, and then context information for each question word (if it appeared). The list of question words was built from the keywords used to identify question types in the pattern matching experiment. This produced 11 words, each of which could have some orthographic variation.

Two content words from the start and from the end of each question were also used as features. Müller (2004) has documented how words at the start of a question are important in question classification, but this work only analysed English questions. In our development set, the average position of question words was the sixth morpheme from the end. 46% of questions had the question word within four morphemes from the end. Importantly, about 5% of the questions occured without a question word, many were fragments, such as:

センデロ	ルミノソ	の
sendero	*ruminoso*	*no*
Sendero	Luminoso	GEN

日本語訳	は	?
nihongoyaku	*ha*	*?*
Japanese translation	TOP	?

"The Japanese translation of Sendero Luminoso is?"

In questions of this form, the words immediately before the topic marker are generally indicative of the EAT.

Both Li and Roth (2006) and Kimura et al. (2005) emphasised the importance of using features that encode linguistic knowledge, especially when the amount of training data is small. The easiest linguistic information to add to the data was the base form of inflected verbs and part of speech tags which were both available from ChaSen. In this experiment, the part of speech was primarily used to filter out non-content words like case markers and verb endings.

The four feature sets used are outlined below, with Feature Set 1 being designed as a baseline experiment, to be directly comparable to the pattern matching experiment since it uses the same information. The other three sets use the POS information from ChaSen to filter out non-content words and then attempt to add semantic information, as used in the Li and Roth (2006) experiment, using whatever linguistic resources were available. Li and Roth used 4 different sources of semantic information: named entity tagging, WordNet, class-specific related words and distributional similarity based categories. In the experiments here Feature Sets 2 and 3 relate to named entity tagging, while Feature Set 4 uses a similar information source to WordNet. The class-specific related words appear to be similar to the lookup table used in the pattern matching experiment, but resources were insufficient to create a similar list large enough to be useful, nor was there a pre-compiled distributional similarity list for Japanese.

Feature Set 1: Word based only

The first set of features used were based on words only (or more precisely, on morphemes according to the ChaSen definition), and hence could be considered to use similar information to that available in the pattern matching experiment. As well as the binary features indicating the presence

ushikubo	*takio*	*san*	*ha*	*genzai*
Ushikubo	Takio	TITLE-HON	TOP	current

saitamaken	*no*	*doko*	*ni*	*dōjō*	*wo*
Saitama	GEN	where	DAT	dojo	ACC

hirai	*te*	*iru*	*ka*
		is operating	QM

"Where in Saitama is Takio Ushikubo currently operating his dojo?"

	Set 1	Set 2
first word:	*ushikubo*	*ushikubo*
second word:	*takio*	*takio*
last word:	*iru*	*hiraku*
second last word:	*te*	*dōjō*
doko **present:**	yes	yes
word *doko* **- 1:**	*no*	`location_rel`
word *doko - 2:*	*saitamaken*	*genzai*
word *doko + 1:*	*ni*	*dōjō*
word *doko + 2:*	*dōjō*	*hiraku*

	Set 3	Set 4
first word:	*ushikubo*	*ushikubo*
second word:	*takio*	*takio*
last word:	*hiraku*	*hiraku*
second last word:	*dōjō*	*dōjō*
doko **present:**	yes	yes
word *doko* **- 1:**	`location_rel`	`location_rel`
type *doko* **- 1:**	PROVINCE	-
word *doko* **- 2:**	*genzai*	*genzai*
type *doko* **- 2:**	-	TIME
word *doko* **+ 1:**	*dōjō*	*dōjō*
word *doko* **+ 2:**	*hiraku*	*hiraku*

Figure 2: An example sentence and the features that would be instantiated for it.

or absence of each question word, the two words either side of each question word were added if the question word was present, as well as the two words at each end of the question (ignoring the question marker if it was present). This gave a total of 59 features, although any one question would normally only have up to 9 instantiated (if only one question word was present).

Feature Set 2: Using SProUT tags

In order to add more semantic knowledge to the data, SProUT (Becker et al., 2002) was used to tag named entities within the question. SProUT uses a fairly small tag set, predominantly tagging names as `person_rel`, `organization_rel` or `location_rel`, and specifying location types where known. A new feature set was created, again recording the presence of question words, and their surrounding context, but if any of the surrounding words had been tagged by SProUT, the SProUT tag was used instead of the word. This has the effect of making, for example, all countries look the same to the classifier, increasing feature counts while ignoring irrelevant information (such as which country).

Feature Set 3: Using Sekine's Named Entity list

A second semantic information source was the gazetteer style list provided with the ENE hierarchy. This list had over 60,000 words and their associated NE type from the taxonomy. For every feature related to a context word or tag in Feature Set 2, another feature was added to record the type of that word if it occurred in the list. That meant the possible number of features went from 59 to 107. Looking through the data, these features were instantiated most often for the names of countries and also for position titles (e.g. President). This feature added more specific named entity information than the SProUT tags, in a form directly related to the possible EATs.

Feature Set 4: Using an Ontology

WordNet (Miller et al., 1990) has often been used as a source of semantic information in English. There is no Japanese version of WordNet, but the Hinoki ontology was available. This ontology was semi-automatically constructed from a Japanese dictionary as described by Bond et al. (2004) and has a fairly limited coverage of about 30,000 words, with very few proper names. This was used to look up each context word in the feature set to see if it was related to any of the named entity types in the ENE. If a related named entity type was found for any word, it was added as a feature, in the same way that NE types were added in Feature Set 3.

An example of how these features were instantiated for a particular sentence is shown in Figure 2.

5 Results

The results for each experiment across all the NE taxonomies are shown in Table 2. These results show that for every NE taxonomy, machine learning techniques were more accurate, with the most accurate results significantly higher than pattern matching for all taxonomies, except NE5. Surprisingly, Feature Set 1, which was based only on words, achieved the best results for the smaller

taxonomies, despite using the same information as the pattern matching. It would be interesting to repeat this experiment, but using POS tags to remove non-content words and see if that gets an even larger improvement. Adding semantic information did not give the expected performance increase, though we begin to see small gains in accuracy due to this extra information as the taxonomies get larger.

While the results show the expected decrease in accuracy as the taxonomies get larger, it is interesting to examine the results in more detail. Breaking down the results by taxonomy, we looked at the classes that were most accurately classified, and those that were difficult to classify. For the NE4 taxonomy the easiest class to classify was NUMBER, correctly identified 94% of the time. THING on the other hand was often misclassified as PERSON, and only correctly labelled in 55% of cases. NE5 differed from NE4 by splitting the PERSON category into PERSON and ORGANIZATION which had the effect of increasing classification accuracy on PERSON, probably because it was now a more tightly defined class. The ORGANIZATION class, however, had very low accuracy (50%), most frequently being misclassified as THING, but also often as LOCATION. This reflects the different ways organisations such as companies can be referred to—in some cases they are places of action and in others, agents of action.

The IREX taxonomy is similar to NE5, but with NUMBER split into DATE, TIME, MONEY, PERCENT and NUMBER. While ORGANIZATION and THING are still mis-labelled in almost half the instances, interestingly, the accuracy for the PERSON and LOCATION classes goes up from the NE5 results. It appears that despite these classes not changing in any way, the feature sets are becoming more discriminatory with the larger taxonomy. Looking at the new numeric classes, DATE and MONEY were the most reliably identified (89% and 94% of instances respectively), but surprisingly PERCENT was only correctly classified 60% of the time, most commonly being mis-labelled as NUMBER. Looking back at the original features, it appears that SProUT tags 何％ *nanpāsento* "what percent" as `percentage_rel`, hence removing the question word 何 *nan* "what" which would otherwise be used as a feature to direct attention to the ％. This is a peculiarity of using SProUT that would need to be addressed in future experiments.

Like IREX, NE15 also has a PERSON, LOCATION and ORGANIZATION class. The differences are that the number classes have been split differently, into NUMBER, TIME and TIME PERIOD, and that THING has been replaced by a number of more specific classes. Classification accuracy for PERSON, LOCATION and ORGANIZATION is almost exactly the same for IREX and NE15. For the number related classes, TIME and NUMBER are detected with accuracies of 86% and 88% respectively. TIME PERIOD is much harder to identify (53% of instances). It is often mis-classified as either TIME or NUMBER. None of the other classes are correctly identified with any great accuracy. Given that THING was only classified with an accuracy of 54%, it is not surprising that subdivisions of THING are not easily identified. Many of the classes have only a few instances in the data set (only EVENT and PRODUCT having over 20) so there are insufficient training examples to allow reliable classification.

NE28 is the hierarchical version of NE15, allowing more specific classes of TIME, NUMBER and TIME PERIOD while still using the more general classes when appropriate. The other differences involve splitting POSITION TITLE out from PERSON, and FACILITY from ORGANIZATION. FACILITY (which is used for things like *school* and *library*) was shown to be very difficult to identify because, like ORGANIZATION, things of type FACILITY can refer to places or entities, and so they were often mis-classified as LOCATION or ORGANIZATION. Most of the number classes were detected with a high accuracy, suggesting that adding the extra specificity for sub-types of NUMBER is a worthwhile modification. The classification accuracy of PERSON was higher again than NE15, possibly because removing the POSITION TITLE type entities tightened the class definition even further.

When the ENE taxonomy was used for classification, 116 out of 148 types were represented, with only 37 types appearing in the gold standard more than 10 times. Not surprisingly, the classification accuracy over the rare types was generally very low, although many of the numeric types were reliably classified despite very little representation in the training data. In general, numbers were used in very specific patterns and this provides good evidence for classification. Examining the types that did occur frequently, it was interesting to note that PERSON was clas-

	NE4	NE5	IREX	NE15	NE28	ENE
P	0.74	0.73	0.71	0.65	0.57	0.41
1	**0.78**	**0.76**	0.73	0.68	0.62	0.48
2	0.77	0.75	0.73	0.68	0.63	0.49
3	0.77	0.75	0.73	0.68	0.63	0.49
4	0.77	0.75	**0.73**	**0.69**	**0.64**	**0.50**

(a) 2000 question set with leave-one-out cross-validation

	NE4	NE5	IREX	NE15	NE28	ENE
P	0.75	0.75	0.69	0.63	0.57	0.42
1	0.73	0.74	0.71	**0.68**	**0.62**	**0.48**
2	0.77	0.75	0.69	0.64	0.61	0.45
3	**0.78**	**0.76**	0.69	0.66	0.60	0.45
4	0.77	**0.76**	**0.72**	0.64	0.61	0.41

(b) Test set from QAC-1

Table 2: Classification accuracy. The first row from both tables has the results from the pattern matching experiment, and the last four show the results from the four feature sets used in machine learning.

	NE28		ENE	
	Exact	Lenient	Exact	Lenient
P	0.57	0.86	0.41	0.62
1	0.62	0.88	0.48	0.60
2	0.63	0.86	0.49	0.60
3	0.63	0.86	0.49	0.59
4	0.64	0.88	0.50	0.60

(a) Leave-one-out cross-validation

	NE28		ENE	
	Exact	Lenient	Exact	Lenient
P	0.57	0.82	0.42	0.67
1	0.62	0.82	0.48	0.70
2	0.61	0.83	0.45	0.64
3	0.60	0.86	0.45	0.65
4	0.61	0.82	0.41	0.60

(b) Test set from QAC-1

Table 3: Comparing exact and lenient evaluations for question classification over the hierarchical taxonomies.

sified more accurately with this taxonomy than any other. The class was more tightly defined in ENE than in some of the smaller taxonomies, but even compared to NE28, which labelled exactly the same entities PERSON, the classification accuracy was 1.5% higher (92.5%). The other classes that were reliably classified were MONEY, many of the time classes, and COUNTRY. The classes that appeared frequently but were often misclassified were generally sub-classes of ORGANIZATION such as COMPANY and GROUP.

5.1 Lenient Evaluation

To look at the effects of the hierarchical taxonomies, a lenient evaluation was performed on the classifications over NE28 and ENE, using the *most likely type* soft decision making strategy described in §2.2. This demonstrates the effective EAT classification accuracy when soft decision making is used in answer extraction. The results, presented in Table 3, show that under lenient evaluation, while ENE accuracy is still quite low, NE28 is very good, better than the accuracy achieved for even the smallest taxonomy.

6 Discussion

A direct comparison with other results is not possible, since published question classification results from this data set could not be found. The Li and Roth (2006) work in question classification by machine learning is the most similar experiment reported, using similar methods but differing in using a different (English) data set, and different taxonomies. Comparing their coarse grained classification results with the accuracy of NE4 and NE5 classifications achieved here, they demonstrated significantly higher results with 92.5%, compared to 78.2% and 75.5% respectively. Taking the NE28 taxonomy as the closest to their fine-grained taxonomy, the performance difference was larger with NE28 classification accuracy at 64.0% being significantly lower than their best result of 89.3%. While the level of difficulty of the two tasks may differ (since the Japanese questions were much longer on average than the English questions), this would not explain such a large performance difference.

There appear to be two main factors that differentiated the experiments. The first obvious difference is the amount of training data. The Li and Roth results reported here were achieved using

21,500 questions for training. In their experiments exploring the effects of training data set size, they found that by reducing the size of the training set to 2000 questions (the amount used here), they lost between 14 and 18% accuracy. While this partially explains the lower results in these experiments, one of the ideas being explored here is the validity of machine learning with small amounts of training data, and hence we are also interested in any other factor that may have led to higher accuracy.

Looking at Li and Roth's detailed evaluation, their most beneficial types of semantic information were the class-specific related word list and the distributional similarity lists, the two forms of information not used here. Their best combination of semantic features (which was the set without WordNet) achieved a 5% accuracy increase over the combination of WordNet and named entity features. It is not clear how the class-specific related word lists were created, except that they were constructed after the manual analysis of 5000 questions. Experiences during the pattern matching experiment suggest that this sort of resource is difficult to build in a general way without overfitting to the questions being analysed. It would be interesting to see whether a translation of Li and Roth's lists would be beneficial to classification on this Japanese question set, or whether they are too closely tied to the set they were created from. Distributional similarity lists however could be built automatically from, for example, the Mainichi newspaper corpus, and could be expected, from Li and Roth's results, to deliver more accurate results.

7 Conclusion

Pattern matching techniques appear to be the basis for the majority of question analysis modules at recent evaluation forums. This is surprising, given that the results here showed that a very basic machine learning based method produced much better classification accuracy with much less effort. Moreover, while pattern matching methods are very language specific (and in our case specific even to the character encoding and tokenisation tool), the machine learning method could be easily used with a new language, provided data was available.

Looking at the results across the different sized taxonomies, predictably the accuracy decreases as the taxonomy size increases. However, looking at the accuracy for specific classes, the larger taxonomies actually led to greater accuracy on some of the more common classes, such as PERSON and COUNTRY, and also on most of the numeric classes, even the less common ones. This may lead to more useful results than high accuracy over very generic classes. Indeed, in further experiments with these taxonomies (reported in Dridan (2007)), the taxonomy that gave the best results for the QA task was the full ENE, even with the classification accuracies reported here.

This raises the question of why we are classifying these EATs, and what are useful distinctions to make for a question answering system. The greatest contribution to inaccuracies in the larger taxonomies was the vast number of infrequent categories that in smaller taxonomies were all classed together as THING. These classes in the ENE appear to be somewhat arbitrary, a criticism that can also be directed towards the fine-grained taxonomy used in Li and Roth (2006). There are no apparent reasons for NATIONALITY being a subclass of ORGANIZATION but RELIGION a top level class of its own, for example, or, in the Li and Roth taxonomy, why MOUNTAIN is a class, but RIVER is not. Often, the classes in fine-grained NE taxonomies appear to be selected for the purpose of classifying questions in the question set they are being used with.

While it might seem a strange argument to suggest, for example, that we use more fine grained numeric classes just because these are the ones we can easily classify, it makes sense to use whatever information is available. Numbers are generally used in very specific patterns in text, and often asked about in very discriminative ways, and these facts should be utilised in an information seeking application. Equally, if it is difficult to classify questions asking about THINGs, there may not be any point to looking for more specific subclasses of THING. This decision will depend on the task at hand, and there may be certain classes of artifact that are important to the application, but arbitrarily creating many subclasses of THING because they were manually identified in a development set is not necessarily productive.

The conclusion to take from this is that large taxonomies can be used for question classification, but more work needs to be put in to determine which type distinctions will actually be use-

ful for question answering, and how they can be arranged in a hierarchy. If different types of numeric data can be accurately classified and detected, for example, then expanding the number of numeric types in a taxonomy is beneficial. This is an interesting area for future research.

References

Markus Becker, Witold Drozdzynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. 2002. SProUT - Shallow Processing with Typed Feature Structures and Unification. In *Proceedings of the International Conference on NLP (ICON 2002)*. Mumbai, India.

Francis Bond, Eric Nichols, Sanae Fujita, and Takaaki Tanaka. 2004. Acquiring an ontology for a fundamental vocabulary. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages pp 1319–1325. Geneva, Switzerland.

Eric Breck, Marc Light, Gideon S. Mann, Ellen Riloff, Brianne Brown, Pranav Anand, Mats Rooth, and Michael Thelen. 2001. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of theACL-2001 Workshop on Open-Domain Question Answering*, pages pp 1–8. Toulouse, France.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2004. TiMBL: Tilburg Memory-Based Learner, version 5.1, Reference Guide. Technical Report ILK 04-02, Tilburg University, Tilburg, The Netherlands. URL `http://ilk.uvt.nl/downloads/pub/papers/ilk0402.pdf`.

Rebecca Dridan. 2007. *Using Minimal Recursion Semantics in Japanese Question Answering*. Master's thesis, The University of Melbourne.

Jun'ichi Fukumoto, Tsuneaki Kato, and Fumito Masui. 2002. Question Answering Challenge (QAC-1): An Evaluation of Question Answering Task at NTCIR Workshop 3. In *Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, pages pp 77–86. Tokyo, Japan.

Yasumoto Kimura, Kinji Ishida, Hirotaka Imaoka, Fumito Masui, Marcin Skowron, Rafal Rzepka, and Kenji Araki. 2005. Three systems and one verifier - HOKUM's participation in QAC3 of NTCIR-5. In *Proceedings of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access*. Tokyo, Japan.

Gakuto Kurata, Naoaki Okazki, and Mitsuru Ishizuka. 2004. GDQA: Graph driven question answering system - NTCIR-4 QAC2 Experiments. In *Working Notes of the Fourth NTCIR Workshop Meeting*, pages pp 338–344. Tokyo, Japan.

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):pp 209–228. URL `doi:10.1017/S1351324905003955`.

Yuuji Matsumoto, Akira Kitauchi, Tatsu Yamashita, and Yoshitake Hirano. 1999. Japanese morphological analysis system ChaSen version 2.0 manual. Technical Report NAIST-IS-TR99009, NAIST, Nara, Japan.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, vol 3(no 4):pp 235–244. URL `http://www.cogsci.princeton.edu/~wn/`.

Karin Müller. 2004. Semi-automatic construction of a question treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*. Lisbon, Portugal.

Satoshi Sekine and Hitoshi Isahara. 1999. IREX project overview. In *Proceedings of the IREX Workshop*, pages pp. 7–12. Tokyo, Japan.

Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. 2002a. Extended named entity hierarchy. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*, pages pp 1818–1824. Las Palmas, Spain.

Satoshi Sekine, Kiyoshi Sudo, Yusuke Shinyama, Chikashi Nobata, Kiyotaka Uchimoto, and Hitoshi Isahara. 2002b. NYU/CRL QA system, QAC question analysis and CRL QA data. In *Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, pages pp 79–86. Tokyo, Japan.

Sentence retrieval for extracting biomedical knowledge

Tara McIntosh and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{tara,james}@it.usyd.edu.au

Abstract

At present, the majority of biomedical Information Retrieval tools process abstracts rather than full-text articles. The increasing availability of full text will allow more knowledge to be extracted with greater reliability. The first step of this is to extract sentences and passages from the text which report scientific results.

We investigate the challenges of sentence retrieval, using an annotated corpus of articles cited in a Molecular Interaction Map (Kohn, 1999) developed by McIntosh and Curran (2007). From the annotated facts we generate keywords for sentence retrieval, and analyse the impact of various query relaxation strategies on performance. We also investigate the impact of *hedging* and *commitment* in the reporting of scientific results on retrieval. Finally, we look at whether linguistic properties such as anaphora and negation have an impact on retrieval performance.

1 Introduction

Almost all known and postulated knowledge relating to biological processes is recorded in the form of semi-structured text, the literature, and data repositories. The volume of biomedical literature rapidly becoming available makes it no longer feasible for biologists to keep abreast of their specialist fields. The standard keyword-based Information Retrieval (IR) approaches over abstracts retrieve too many articles that must be manually inspected.

There is considerable interest in incorporating NLP tools into IR systems to overcome this information bottleneck. There is also a strong focus on improving Information Extraction (IE), which attempts to identify the relationships, such as interactions, between bio-entities, including genes and proteins (Bunescu et al., 2006).

The primary biomedical IR tool, the National Library of Medicine's PubMed, allows researchers to search for relevant documents using keyword-based queries over abstracts. Research systems are beginning to bring NLP techniques to bear on this task. The MEDIE system identifies biomedical relationships within individual sentences in Medline abstracts, which are first parsed using the HSPG grammar (Ohta et al., 2006). As McIntosh and Curran (2007) showed, a significant number of interactions are not contained within the abstract of a paper. There is a need for IR/NLP tools to exploit the growing number of publicly available full-text articles.

Since NLP tools are currently quite slow, an important step in this process is to identify sentences and passages that are likely to contain scientific results, which is basically an IR task at the sentence level. Also, referring scientists to particular sentences will reduce the burden of reading full-text articles to identify interactions.

In this paper, we analyse some factors influencing the sentence retrieval problem for biomedical fact extraction from full-text articles. We use a corpus of full-text articles that have been exhaustively annotated with molecular interactions (McIntosh and Curran, 2007). The corpus is based on the *Molecular Interaction Map* (MIM), constructed by Kohn (1999), and documents the process of inferring the MIM facts from full text. Since we have exhaustively identified and annotated all of the sentences supporting these specific facts, we can reliably identify all relevant and irrelevant retrieved sentences under a range of conditions.

We evaluate various keyword queries with respect to their ability to identify previously annotated positive sentences describing molecular interactions in the full-text corpus. In the first set of

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 342–349,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

experiments, simple keyword queries were generated from the description of the facts in the MIM by a domain expert. The keywords fall into various classes, e.g. the entities involved in a particular interaction or the verbs describing the interaction. We then explored various forms of query relaxation, and their impact on sentence retrieval.

Next, we investigate the impact of *hedging* and *commitment* on sentence retrieval. Hedging is used to indicate uncertainty or scepticism about a particular proposition (Hyland, 1996). For example, the following sentences express the same proposition between two proteins `RPA1` and `DNA-PK`, however only the first does so with certainty:

1. RPA1 was sufficient to form a complex with DNA-PK.
2. Experiments were performed to test whether DNA-PK could form a protein complex with RPA1.

Hedging has been studied in the citation analysis of scientific literature (Mercer and Marco, 2004). Our study is the first we are aware of to investigate the impact of hedging on sentence retrieval. We compare the frequency of hedging and commitment words in the output of the system.

Sentences in the MIM corpus were also annotated with various linguistic properties, e.g. whether it requires the resolution of anaphora or negation for the fact to be inferred from the sentence (McIntosh and Curran, 2007). We investigate what proportion of the results contain these various phenomena.

Our conclusions are that some levels of query relaxation are required to reach acceptable levels of fact recall (although the cost in precision is quite high); that hedging and commitment terms appear frequently in both relevant and irrelevant results; and that even the most relaxed queries fail to retrieve a significant proportion of sentences requiring anaphora, negation and extra fact resolution.

2 Biomedical IR and NLP

At present, most biomedical IR and IE tools process abstracts rather than full-text articles (Ohta et al., 2006). This is due to the availability of abstracts (Hirschman et al., 2002) and the lack of full-text training corpora. The majority of bioNLP corpora consist of sentences or whole abstracts annotated for biomedical Named Entity Recognition (NER) and IE, such as GENIA (Kim et al., 2003) and BioInfer (Pyysalo et al., 2007). For other bioNLP tasks, such as coreference resolution, there is very limited training data. The BioInfer corpus is also annotated with coreference expressions, however they do not annotate those which cross sentence boundaries (Pyysalo et al., 2007).

Full-text articles are becoming increasingly available to NLP researchers, who have identified the importance of processing specific full-text sections. Regev et al. (2002) developed the first bioIR system specifically focusing on limited text sections, primarily Figure legends. Their performance in the KDD Cup Challenge showed the importance of considering full-text article structure. Yu et al. (2002) showed that the Introduction defines the majority of synonyms, while Schuemie et al. (2004) and Shah et al. (2003) showed that the Results and Methods are the most and least informative, respectively. In contrast, Sinclair and Webber (2004) found the Methods useful in assigning Gene Ontology codes to articles.

Full-text articles also have the advantage of repeating facts in different contexts, increasing the likelihood of an imperfect IR system identifying them (McIntosh and Curran, 2007). This redundancy can also be used for passage validation and ranking (Clarke et al., 2001).

Hedging is frequently used in scientific literature to indicate a lack of commitment to a statement, scepticism, and/or open-mindedness about propositions (Hyland, 1996). For example, the terms *may, might, propose*, and *possibly* convey speculation, compared to *does* and *indicates*. Hedging occurs more frequently in citation contexts than in the text as a whole (Mercer and Marco, 2004). Light et al. (2004) introduced three levels of certainty, highly speculative, low speculative and definite, in biomedical literature. In this work, we expand the list of speculative words described in the literature to automatically analyse their impact on sentence retrieval.

3 Full-text MIM corpus

McIntosh and Curran (2007) manually extracted and annotated a corpus of sentences or short passages from full-text articles. Each annotated sentence, known as an *instance*, expresses a documented interaction between bio-entities. The interactions identified are based on the *Molecular Interaction Map* (MIM) constructed by Kohn (1999), which describes 203 different interactions

1.	N4 Main fact: *RPA2 binds XPA via the C-terminal region of RPA2*
TP	Mutant RPA that lacked the p34 C terminus failed to interact with XPA, whereas RPA containing the p70 mutant (Delta RS) interacted with XPA (Fig. 2). (Results)
TP	We found that the C-terminal domain of RPA p34 is responsible for RPA interaction with XPA. (Intro.)
FP	For this, we compared the XPA-DNA interaction of wild-type RPA with that of mutants lacking either the N-terminal or C-terminal domain of p34 (RPA:p34Delta 2-30 and RPA:p34Delta 33C, respectively) (Fig. 4). (Results)
FP	Once XPA and RPA form a stable complex on the DNA, they are thought to bring other repair proteins to the site of initiation of nucleotide excision repair. (Discussion)

Table 1: Example TP and FP retrieved for a Main fact matching keywords in Table 3

between bio-entities in mammalian cells. Each interaction in the MIM is associated with a description that summarises the evidence for the interaction from the literature, including the relevant citations.

Each main concept in the summaries is referred to as a *main fact* or a *subfact*, which represent part of a main fact, in the corpus. Text substantiating these facts from the cited articles were manually retrieved and annotated in the corpus. For example, instances in the corpus supporting the main fact N4 *RPA2 binds XPA via the C-terminal region of RPA2* in Table 1 are indicated with the TP (true positive) tag. The two instances in Table 1 were identified in the Results and Introduction sections.

The corpus contains 1637 annotated instances which support 77 of the 203 MIM summaries in Kohn (1999), totalling 1738 sentences from 64 full-text HTML articles. Only 92 instances consist of two or more adjacent sentences, which are all needed to substantiate a fact, e.g. because an anaphor's referent is in the previous sentence. Each instance has been annotated with linguistic phenomena, such as coreference and negated expressions which needs to be resolved for the automatic extraction of the exact relationship mentioned in the MIM fact. Instances which also depend on additional knowledge from the full text which is necessary to infer the MIM fact are also annotated. These logical dependencies include synonyms and extra facts, which may or may not be defined in the same article. For example, the two TP in Table 1 require the resolution of the synonymous bio-entities `RPA2` and `p34`, to infer the main fact from the text. Full details of this corpus are detailed in McIntosh and Curran (2007).

The IR experiments reported here are based on the set of 64 full-text articles. The articles were converted from HTML to plain text using Lynx followed by some manual post-processing to filter out any remaining noise. We used a boundary detector based on the MXTerminator (Reynar and Ratnaparkhi, 1997) with modifications to handle common mistaken boundaries such as `et al.` These sentences were tokenised, ensuring entities with punctuation, like `E2F-4` were not split. The 64 articles contain 16,257 sentences and 363,131 words.

4 Full-text IR

The experiments we describe here are based on a IR system, which retrieves all sentences, without ranking, matching a specific set of keywords in a full-text article. Each search is restricted to a particular article (or articles) which is known in advance to contain text supporting the relevant MIM facts. The task of the IR system is to retrieve just those sentences annotated as positive instances, based on keywords created from the description of the fact by a domain expert.

4.1 Evaluation

For a given MIM fact, any retrieved sentence that matches the keyword queries, which also appears as an annotated instance for the specific fact in the corpus is considered a relevant result, that is, a true positive (TP). Since McIntosh and Curran (2007) did an exhaustive manual search for sentences supporting a given fact, any other sentence retrieved by the system is irrelevant, that is, a false positive (FP). Finally, any annotated instances that were not retrieved by the system are false negatives (FN).

For the keyword expansion results we also report precision P, recall R and F-score F:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$R = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$F = \frac{2PR}{P + R}$$

2.	A5 Main fact: *c-Abl phosphorylates tyrosines in the C-terminal domain of RNA polymerase II*
FP	`Given the fact` that Arg and Abl are highly divergent in the C-terminal region except for the CTD-interacting domain, it is `possible` that these two kinases `may` transduce different signals to mediate the tyrosine phosphorylation of RNA polymerase II. (Discussion)
3.	B3 Subfact: *DNA-PK can bind dsDNA without Ku*
FP	`It seems likely` that DNA-PK does not recognize DNA alone initially if Ku is present; rather it binds to `some part` of Ku or both Ku and DNA in the Ku:DNA complex. (Discussion)
4.	C43 Main fact: *p16 associates with TFIIH and RNA pol II CTD*
FP	`The possibility` that p16[INK4A] `might` associate with the RNA pol II, a protein substrate of the CTD kinase of TFIIH, was `next examined`. (Results)

Table 2: Example FP containing hedging terms

4.2 Keywords and Queries

For each of the facts in the corpus, we generated keyword lists for the main terms associated with the instances. These lists were created semi-automatically by first obtaining the most frequent terms from the instances, excluding stop words.

Each list was manually reduced by a domain expert to only those associated with the fact and separated into the three classes: *bio-entities* involved in the interaction including synonyms, *verbs* describing the interaction, and *extra terms* which were considered necessary by the domain expert to fully identify the entire fact. Extra terms typically refer to specific structures within the entities involved in the interaction, and in many cases no extra terms were specified by the domain expert.

For example, Table 3 shows the keyword list for the MIM fact in Table 1. The synonyms are in parentheses. The bio-entity `p70` in the first TP instance for this MIM fact (Table 1) is not included in the list because it isn't part of the fact.

These classes of keywords are then used to form queries with various levels of relaxation. The verb class is relaxed in two ways, giving a total of five keyword classes:

ent sentences must contain all main entities

verb sentences must contain all main verbs associated with a MIM fact

verb syn as above, but sentences may contain synonyms for each main verb

any verb sentences must contain at least one verb from the set of all main verbs and synonyms

extra if a MIM fact is associated with extra terms, sentences must contain these or their synonyms

These five classes are combined together to form the query sets described in Section 5.

Terms	Keywords
entities	RPA2 (p34, RPAp34), XPA
verb	binds (associates, complex)
extra term	C-terminal (carboxyl-terminus)

Table 3: Search keywords for fact N4

4.3 Hedging and Commitment

Scientific literature contains various levels of certainty, ranging from doubt to complete confidence, and *hedging* is frequently used to indicate any lack of commitment to a fact (Hyland, 1996). Hedging is typically realised using modal verbs, epistemic adjectives, nouns and adverbials, lexical verbs and indefinite quantifiers (Holmes, 1988). For the analyses reported here, we experimented with terms in each of these categories identified by Holmes (1988) expressing uncertainty, and the small set of speculative terms identified by Light et al. (2004). We also constructed a list of commitment terms – these words express a high degree of certainty. These were manually identified in the corpus by a domain expert whilst they were performing an analysis of the false positives and negatives. Examples of each of these lists are shown in Table 6.

Examples 2–4 in Table 2 are FP of their respective facts which contain at least one hedging expression. In example 2, the epistemic adjective *possible* and modal verb *may* are used to express open-mindedness of the fact. Epistemic adjectives and modal verbs are also used in examples 3 and 4 respectively. In examples 2–4, the authors are indicating their research aims and hypotheses. However, this is only directly stated in example 4, indicated by the phrase *was next examined*. If we ignore the notion of hedging, only examples 2 and 3 would match the MIM facts. Although example 4 contains all entities and the exact verb used to describe their relationships (*associate*), it only

5.	P21 Subfact: *p300 acetylates p53*
FN	Note that incubation with DNA-PK produced a new `p53 isoform` (labeled 3) that is phosphorylated on Ser-37 as well as on Ser-33. `This isoform` was preferentially acetylated by p300. (Figure legend)
6.	A4 Subfact:*c-Abl inhibits Mdm2-mediated degradation of p53*
FN	We demonstrate that `c-Abl increases the expression level of the p53 protein.` The `enhanced expression` is achieved by inhibiting Mdm2-mediated degradation of p53. (Abstract)
7.	N6 Subfact: *XPF binds to the C-terminal region of ERCC1*
FN	Previous mutagenesis studies showed that a 'Rad10-like' `ERCC1 protein, with a stop at residue 214,` was functionally inactive (27). This can now be explained by the `inability of this protein to form a complex with XPF`. (Discussion)
8.	P36 Subfact: *TBP binds to an acidic domain in central Mdm2*
FN	We show that MDM2 binds to the general transcription factor TFIID in vivo. The C-terminal Ring finger interacts with TAF[II]250/CCG1, and the central acidic domain interacts with TBP. (Abstract)

Table 4: Example FN containing coreference expressions

substantiates *part* of the MIM fact (*p16 associates with RNA pol II*).

4.4 Linguistic Phenomena

McIntosh and Curran (2007) annotated the instances in the MIM corpus with linguistic phenomena and extra fact dependencies. We use these annotations to evaluate the potential impact on sentence retrieval of preprocessing the text with NLP tools that interpret these phenomena.

Coreference expressions are often used in biomedical literature to make abbreviated or indirect references to bio-entities or processes. *Negated* expressions include descriptions of an abnormal condition, such as experimental mutations, and the resulting abnormal outcome, such as cancer. From these negated expressions a normal condition and outcome can be inferred.

Examples of these are shown in Table 4. Unlike the hedging examples (Table 2), each of these sentences expresses the authors' complete confidence in the facts. Although each of these examples contains two sentences, the keyword searches are unable to identify either of the sentences. This is because no single sentence contains all of the keywords associated with the fact.

Examples 5–7 require anaphora resolution, e.g. example 6 involves linking the event anaphor `enhanced expression` to the function of `c-Abl` in the first sentence (antecedent), and then the role of `c-Abl` in the inhibition can be inferred.

Example 7 is complicated further by the negated expressions. The anaphor `this protein` refers to the mutated/truncated form of the protein `ERCC1` (with a stop at residue 214), rather than `Rad10` or normal `ERCC1`. Two negative expressions must then be processed. First the mutated form of `ERCC1` is `inactive`, and the second negated expression states that the mutated `ERCC1` is unable to bind `XPF`. From resolving and merging these two negatives we can infer the MIM subfact of N6.

In example 8, the first sentence states a different interaction to that in the MIM fact, and there is no specific coreference expression linking these sentences together, however to infer the fact the reference of the `C-terminal ring finger` needs to be associated with the bio-entity `MDM2`.

	P	R	F	TP	FP	FN
ent + verb + extra	34	35	34	601	1168	1137
ent + verb	28	37	32	647	1640	1091
ent + verb syn + ext.	33	63	43	1089	2188	649
ent + verb syn	26	67	37	1165	3342	573
ent + any verb	14	73	24	1274	7582	464
ent	13	75	22	1300	8932	438

Table 5: IR performance for various queries

5 Results and Discussion

Table 5 shows the precision (P), recall (R) and F-score (F), and the distribution of TP, FP and FN results for sentences matching keyword queries with varying specificity for individual MIM facts.

The first experiment in Table 5 uses the most restrictive queries, requiring all of the keywords: bio-entities, main verbs and extra terms to be present. This query is unrealistic because it requires knowledge of the exact relationship being known in advance – the exact interaction verbs and extra terms. As a result it achieves the highest precision of 34% but the lowest recall of 35%.

Each subsequent experiment shown in Table 5 relaxes the search criteria, and so recall increases

Word list	TP	FN	FP	Examples
Epistemic adjectives	2.44	2.45	4.58	probable, possible, unlikely
Epistemic nouns	2.29	2.14	3.98	chance, claim, suggestion
Modal verbs	8.32	9.48	15.04	could, should, might
Epistemic adverbials	2.52	2.45	4.06	maybe, perhaps, presumably, surely
Indefinite quantifiers	3.59	6.73	5.27	about, generally, often, sometimes
Epistemic lexical verbs	12.67	16.21	18.15	appear, hypothesize, presume, suggest
Speculative words (Light et al., 2004)	13.05	14.07	16.08	likely, may, suggest, promise
Any hedging word	24.96	31.80	37.51	
Any positive word	40.31	48.01	36.65	demonstrate, established, indicating
Only hedging words	15.34	15.29	24.89	
Only positive words	30.69	31.50	24.03	

Table 6: Hedging and Commitment

and precision decreases. The least restrictive search, *ent*, results in the largest recall and the worse precision, returning an enormous number of FP. Unfortunately, these are exactly the kinds of queries a biologist might initially type into PubMed.

There is a significant improvement in F-score, from 32% to 37%, when the corresponding verb lists are expanded to include their synonyms (*ent + verb syn*), but the number of FP increases by 51%.

The best performance of 43% F-score is achieved with the *ent + verb syn + extra* queries, that is the bio-entities, the verbs or their synonyms and the extra terms. Including the extra terms significantly reduces the number of FP, however, this search unrealistically relies heavily on prior knowledge of the exact MIM fact. Unfortunately, the most realistic query setting is *ent + any verb*, since it is feasible to enumerate possible interaction verbs and their synonyms without prior knowledge of the type of interaction.

5.1 Hedging and Commitment

The results of our hedging and commitment experiments are shown in Table 6. This table shows the importance of the various categories of hedging expressions, detailed in Holmes (1988) and Light et al. (2004), as well as the words expressing certainty and definite facts (positive words). Example terms for each of these categories are also shown in Table 6.

We are interested in identifying any potential IR improvement that may be gained from recognising hedging and commitment in articles. More specifically, can hedging and or commitment be used to help separate relevant and irrelevant sentences.

The majority of hedging categories occur in less than 5% of TP, FN and FP. The most significant class discrimination is obtained with modal verbs, with a high 15.04% of FP containing at least one, with almost 7% difference between the TP and FP. Epistemic lexical verbs and the speculative words identified by Light et al. (2004) are the least discriminative, and are also very frequently occurring in the literature. We then investigated the overall importance of hedging words by combining the hedging categories into one word list, and identified those TP, FN and FP which contain any hedging terms. These results indicate that hedging cannot be used as a form of filtering of FP, as a high proportion of both TP and FN also contain hedging words.

We then analysed terms expressing commitment in the retrieved sentences. As expected many of the TP and FN contain positive words, 40.31% and 48.01% respectively, however a large proportion of FP (36.65%) also contain positive language. These experiments do not consider the possibility of both hedging and commitment terms appearing in the same sentences. For instance, it is common for scientists to present a known fact with commitment, and then speculate about possible reasons or future studies, for example:

While DNA-PK clearly interacts with DNA on its own, based on the DNA-PK activation by DNA alone (Fig. 1), the nature of the stimulation by Ku is still unclear.

When we consider sentences with hedging words and no commitment words (Only hedging words) and vice versa, the FP are separated more from the TP and FN.

	Negated	Anaphora	Event Ana.	Cataphora	Extra Dep.	None
TP & FN Instances	101 (%)	145 (%)	34 (%)	23 (%)	561 (%)	892 (%)
ent + verb + extra	60 (59.4)	93 (64.1)	30 (88.2)	10 (43.5)	358 (63.8)	450 (50.4)
ent + verb	60 (59.4)	88 (60.7)	30 (88.2)	10 (43.5)	338 (60.2)	434 (48.7)
ent + verb syn + extra	25 (24.8)	59 (40.7)	24 (70.6)	6 (26.1)	209 (37.3)	120 (13.5)
ent + verb syn	24 (23.8)	53 (36.6)	24 (70.6)	6 (26.1)	177 (31.6)	95 (10.7)
ent + any verb	20 (19.8)	49 (33.8)	22 (64.7)	6 (26.1)	140 (25.0)	67 (7.5)
ent	18 (17.8)	46 (31.7)	21 (61.8)	6 (26.1)	135 (24.1)	49 (5.5)

Table 7: Distribution of linguistic phenomena in FN instances

Our analysis shows that hedging and commitment terms cannot be used for filtering FN on their own, however these categories, in particular the modal verbs, may be used in the development of models with other features to distinguish between TP and FP.

5.2 Linguistic Phenomena

We are interested in understanding why certain FN are not been identified. For each FN we investigated which linguistic phenomena may be responsible for them going undetected. Table 7 details the linguistic phenomena associated with the instances in the MIM corpus. The second row shows the total number of instances annotated with the specific phenomena. The main linguistic phenomena that are annotated in the corpus, which require resolution are extra fact dependencies (561), followed by anaphoric expressions (145). Cataphoric expressions are not common with only 23 instances requiring these to be resolved. Some instances may be annotated with multiple linguistic constructs. The majority of the instances (892) do not contain any of these characteristics.

The majority of instances which contain annotated phenomena are identified with the most relaxed search (*ent*). For instance only 46 of the FN contained at least one anaphoric expression, and this accounts for 31.7% of all instances annotated with anaphora. Many of these FN contain more than one sentence, where no single sentence mentions each of the entities. Similar results are observed with event anaphora, with 61.8% of instances going undetected. Examples of these are shown in Table 4.

The next main issue is the notion of extra fact dependencies, and many instances (135) requiring these to be resolved are also undetected with the most relaxed search. In McIntosh and Curran (2007) both synonym and extra fact dependencies indicated the necessity for full-text processing and merging of knowledge throughout the article.

Considering the F-score associated with the two least restrictive searches, in particular the extremely large number of FP identified (Table 5) it is more realistic to look at those obtained with the *ent + verb syn* query. There is overall a slight increase in the FN requiring linguistic phenomena to be resolved.

6 Conclusion

In this paper, we have presented an analysis of biomedical keyword-based full-text IR, investigating the characteristics of various query relaxation strategies, hedging and commitment, and identification of linguistic phenomena which may improve relevance.

Our first experiments show that the most restrictive search with query terms matching those in the actual MIM facts is not successful. This is due to the high level of synonymous verbs in biomedical texts. Furthermore, searching only for bio-entities results in too many FP. The best F-score of 43% is achieved with the *ent + verb syn + extra* query.

Hedging has previously been reported to be most commonly used when citing in scientific literature. In this study, we investigated the impact of hedging on the reporting a new scientific findings. We find that while hedging appears frequently in irrelevant sentences, it also appears frequently in relevant sentences. The most discriminative class of hedging was the modal verbs. We also considered commitment terms used in scientific literature, however these also were commonly used across TP and FP. Therefore, hedging and commitment terms cannot be used as a means of post-processing and filtering.

Linguistic analysis of FN shows the potential improvements gained from anaphora resolution and the identification of extra fact dependencies.

Even with the most relaxed queries, relevant sentences are often missed due to these characteristics. Furthermore, with the most reliable query level, *ent + verb syn + extra*, these phenomena are significantly more important to resolve.

In summary, this paper provides guidance to developers of biomedical IR systems that operate on the sentence level. In future work, we will be implementing a system that incorporates the kind of NLP required to exploit the linguistic phenomena identified as important here.

Acknowledgements

This work was supported by the CSIRO ICT Centre and ARC Discovery grants DP0453131 and DP0665973.

References

Razvan Bunescu, Raymond Mooney, Arun Ramani, and Edward Marcotte. 2006. Integrating co-occurrence statistics with information extraction for robust retrieval of protein interactions from medline. In *Proceedings of the BioNLP Workshop on Linking Natural Language Processing and Biology at HLT-NAACL 06*, pages 49–56. New York City.

Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 358–365. New Orleans, LA.

Lynette Hirschman, Jong C. Park, Junichi Tsujii, Limsoon Wong, and Cathy Wu. 2002. Accomplishments and challenges in literature data mining for biology. *Bioinformatics Review*, (12):1553–1561.

Janet Holmes. 1988. Doubt and certainty in ESL textbooks. *Applied Linguistics*, 9(1):21–44.

Ken Hyland. 1996. Writing without conviction? Hedging in science research articles. *Applied Linguistics*, 17(4):433–454.

Jin-Dong Kim, Tomoko Ohta, Yuka Teteisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(1):i180–i182.

Kurt W. Kohn. 1999. Molecular interaction map of the mammalian cell cycle and DNA repair systems. *Molecular Biology of the Cell*, 10:2703–2734.

Marc Light, Xin Ting Qui, and Padmini Srinivasan. 2004. The language of bioscience: Facts, speculations, and statements in between. In *Proceedings of BioLink 2004 Workshop on Linking Biological Literature, Ontologies and Databases: Tools for Users*, pages 17–24. Boston.

Tara McIntosh and James R. Curran. 2007. Challenges for extracting biomedical knowledge from full text. In *Proceedings of the Workshop on BioNLP 2007, Biological, Translational, and Clinical Language Processing*, pages 171–178. Prague, Czech Republic.

Robert E. Mercer and Chrysanne Di Marco. 2004. A design methodology for a biomedical literature indxexing tool using the rhetoric of science. In *Proceedings of BioLink 2004 Workshop on Linking Biological Literature, Ontologies and Databases: Tools for Users*, pages 77–84. Boston.

Tomoko Ohta, Yusuke Miyao, Takashi Ninomiya, Yoshimasa Tsuruoka, Akane Yakushiji, Katsuya Masuda, Jumpei Takeuchi, Kazuhiro Yoshida, Tadayoshi Hara, Jin-Dong Kim, Yuka Tateisi, and Jun'ichi Tsujii. 2006. An intelligent search engine and GUI-based efficient Medline search tool based on deep syntactic parsing. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 17–20. Sydney, Australia.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Bjorne, Jorma Boberg, Jouni Jarvinen, and Tapio Salakoski. 2007. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(50).

Yizhar Regev, Michal Finkelstein-Langau, Ronen Feldman, Mayo Gorodetsky, Xin Zheng, Samuel Levy, Rosane Charlab, Charles Lawrence, Ross A. Lippert, Qing Zhang, and Hagit Shatkay. 2002. Rule-based extraction of experimental evidence in the biomedical domain - the KDD Cup 2002 (Task 1). *ACM SIGKKD Explorations*, 4(2):90–92.

J.C Reynar and A. Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 16–19. Washington DC.

M.J. Schuemie, M. Weeber, B.J.A. Schijvenaars, E.M. van Mulligen, C.C. van der Eijk, R.Jelier, B.Mons, and J.A Kors. 2004. Distribution of information in biomedical abstracts and full-text publications. *Bioinformatics*, 20(16):2597–2604.

Parantu K. Shah, Carolina Perez-Iratxeta, Peer Bork, and Miguel A. Andrade. 2003. Information extraction from full text scientific articles: where are the keywords? *BMC Bioinformatics*, 4(20).

Gail Sinclair and Bonnie Webber. 2004. Classification from full text: A comparison of canonical sections of scientific papers. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA)*, pages 66–69. Geneva, Switzerland.

Hong Yu, Vasileios Hatzivassiloglou, Carol Friedman, Andrey Rzhetsky, and W.John Wilbur. 2002. Automatic extraction of gene and protein synonyms from Medline and journal articles. In *Proceedings of the AMIA Symposium 2002*, pages 919–923. San Antonio, TX.

Centering in Chinese Anaphor Resolution: A Parametric Study

Duan Manjuan
Shanghai International Studies University
550, Dalian Road (W), Shanghai
200083, China
julianneduan@hotmail.com

Abstract

The basic concepts in Centering Theory, such as utterance, ranking and R1-pronouns, are purposely left unspecified to concrete tasks. The proper setting of these parameters in Chinese anaphoric resolution is examined in this study. Six resolution algorithms are designed to examine the effectiveness of the instantiations of different parametric settings. The results reveal that utterances, when segmented into smaller chunks, are more suitable for zero pronoun resolution (ZP resolution). Pronoun resolution (PRON resolution) shows relatively lower degree of sensitiveness to the change of utterance definitions. The ranking-affecting factors, grammatical functions and syntactic hierarchies, have greater bearing on the resolution. The influence from the parallelism of grammatical functions is testified but marginal in many instantiations. The C_b continuity claimed by Centering demonstrates weak correlation with the antecedent determination. The pervasive higher precision rate in ZP resolution suggests that zero anaphors are more suitable to be chosen as R1-pronouns in Centering model.

1 Motivations

Centering Theory has proven to be an effective and useful conceptual framework for theorizing coherence and salience in local discourse and has been widely referred to in anaphoric resolution in many languages. However, the theoretical concepts, such as "utterance", "realization" and "ranking", which serve as the basis of the theory, are left unspecified intentionally. The researchers working on Centering argue that while these concepts play a central role in any theory of discourse coherence and salience, their precise characterization is best left for subsequent specific research. Those notions, as Poesio et al. (2004) concludes, are the parameters of Centering.

This research is an empirical study exploring the optimal setting of these Centering parameters in Chinese anaphora resolution. We hope, though a systematic investigation on the parametric setting of Centering, we can acquire a deeper insight into the discourse salience and coherence, and also provide practical implications for Chinese anaphora resolution.

2 Centering Theory and its Parametric Instantiations

Centering Theory (Grosz et al.1995) is a model of the attention state in discourse and its inter-relationship with inferential complexity and form of referring expressions. Each utterance U in a discourse segment is assigned a set of forward-looking centers, $C_f(U)$; each utterance other than the segment initial utterance is assigned a single backward-looking center, $C_b(U)$. The backward-looking center of U_{n+1} connects with one of the forward-looking centers of U_n.

The highest ranked member of the set of forward-looking centers is referred to as the preferred center, C_p. The preferred center represents a prediction about the C_b of the following utterance, but this is not necessarily the case.

Centering model also includes a set of rules and constraints. Two Centering rules are Rule1and Rule 2. Rule 1 stipulates that if any element of $C_f(U_n)$ is realized by a pronoun in U_{n+1}, then the C_b (U_{n+1}) must be realized by a pronoun also. It is the main claim of Centering about a preference for pronominalizing the C_b, if anything is pronominalized at all. The two versions of Rule 1 are proved to be very robust in Poesio et al. (2004). However, as noticed by McCoy and Strube (1999), Rule 1 applies only in the cases where two subsequent utterances share more than one referent, and the non-C_b referent is pronominalized. But why this non-C_b referent is

Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 350–358,
This work is licensed under a Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/).

realized as a pronoun is left unexplained by the theory.

Nevertheless, following more the spirit of Centering than the actual definition, we can understand the C_b, as a referent which is preferably pronominalized, actually stands for the most prominent discourse entity of the previous utterance. Pronominalization of the C_b coincides with the common hypothesis that the more salient the discourse entity is, the more likely it is to be pronominalized. The mismatch between the C_b and the pronominalization may illustrate the situation that by having been mentioned in a discourse often, a discourse entity may become sufficiently salient to justify pronominalization, even when it is not the C_b. These observations suggest that the decision to pronominalize does not depend only on whether a discourse entity is the C_b, but also involves a number of further constraints and preference to justify its pronominaliztion.

Rule 2 is a rule of preference among transition types, i.e., sequences of continuation are preferred over sequences of retaining; and sequences of retaining are preferred over sequence of shifting (Grosz et al.1995). It attempts to characterize entity-coherent discourse: discourses segments in which successive "utterances" keep mentioning the same discourse entities are "more coherence" than discourse segments in which different entities are mentioned. Centering Theory states an obvious preference for preserving the C_b over changing it, and for preserving it as the most salient entity over changing its relative ranking.

In practical application of Centering to anaphoric resolution algorithms, only BFP (Brennan, Friedman and Pollard. 1987) resorts to Rule 2 faithfully and COT (Centering in Optimization Theory) (Beaver 2004) uses a decomposed Rule 2 in an optimal way. Other Centering-based algorithms do not include Rule 2 as a constraint in antecedent determining, since, as Kehler (1997) holds, Rule 2's role in pronoun resolution is claimed to be unknown yet. Accordingly, in this research we do not take Rule 2 as a determining factor in antecedent choosing, but its implication is examined as a ranking-affecting factor in the form of C_b continuity.

2.1 Centering Update Units

Defining the update unit within the framework of Centering model is central because it is closely related to the appropriate description of the changing focus state. Given that anaphoric expressions occur in all types of clauses, it was crucial that the size of the unit was constrained to enable the handling of intra-sentential anaphora and at the same time account for the Centering shift among the utterances. It is desirable if the definition of utterance can reflect our intuition about perceived discourse sequence and the processing load required by the reader/hearer at any given moment in discourse processing.

In view of the rather weak inflectional features and the always absence of conjunctives in Chinese, we examine two variations of utterance parameter in current study: 1) utterance identified with the discourse sequence which contains at least one predicate structure and demarked by punctuations like commas, colons, semicolons or other sentence-terminal punctuations (Udef.1); 2) Utterance identified with the discourse sequence demarked only by sentence-terminal punctuations (Udef.2). They are full stops, exclamatory marks and question marks.

Udef.1 is widely employed for the fact that comma and other non-sentential-terminal punctuations in Chinese are most frequently used to signal pauses within sentences (Yeh and Chen 2003; Wang 2004; Xu 2003). In fact, Udef.1, to some extend, shares Kameyama's propose of breaking complex sentence into clauses, for both try to reduce the computational load by breaking discourse into relatively shorter discourse chunks.

Udef.2 is inspired by Suri and McCoy (1999) and Poesio et al. (2004), which hold that adjunct clauses are best processed as embedded and that the larger the utterance is defined, the less violations of Constraint 1 occur. Although sentences in Chinese are different from those in English by nature, we believe the sentence terminal punctuations are always used for their reason to wrap up the information in current chunk and star a new segment.

2.2 Ranking

The set of forward-looking centers, C_f, is ranked according to discourse salience. Although in most Centering literatures grammatical roles of the discourse entities in C_f is taken as the default ranking criteria, the advocators of Centering clearly suggest that the ranking of looking-forward centers is also open to the influence of other factors, such word order (especially fronting), clausal subordination, as well as the interaction among these factors.

- Linear order vs. grammatical functions

Based on the ranking scale of grammatical roles proposed by Brennan et al. (1987), which distinguishes direct objects and indirect objects, Wang (2004) and Yeh and Chen (2003), which consider topic prominence in Chinese, and Jiang (2004), which proposes the separate consideration of NP modifiers in salience ranking, we divide our salience rank into three levels: 1) constituents of subjecthood, like topic, subject and predicate NP in existential construction, with topic ranking higher for its stronger prominence; 2) constituents of objecthood, like focus constituent[1], object and indirect object, with focus ranking higher; 3) constituents of adjuncthood like oblique and adverb, followed by their modifiers. Therefore the grammatical function scale assumed in this study finally come out as: TPC[2]/SBJ/ EPRDN> TM/ SM/ EM> FOC/OBJ/ IO> FM/ OM/ IM/ AM/ PM> OBL/ADV> OBLM/ ADVM/ APPM. Under the situation where there is more than one discourse entity assigned the same salience in terms of their grammatical functions, the occurring order is used as a tie-breaking factor to achieve a total ranking of C_fs.

Linear order of the discourse entities is also examined in current research as ranking criterion. Rambow (1993) holds that ranking in German is to be determined by surface order of realization. The results of probe experiments by Corbett and Chang (1983) suggest that order of mention affects recall from memory, and in particular, that the first-mentioned discourse entity in a sentence is the most salient. Gordon et al. (1993) observed a repeated name penalty (RNP) for C_fs in subject position, suggesting that first mentioned C_fs are as highly ranked as subjects.

- C_b Continuity vs. Syntactic Parallelism

C_b continuity is designed to test the predicting effectiveness of the "entity coherence" in antecedent determination, which states a preference for preserving the C_b over changing it, and for preserving it as the most salient entity over changing its relative ranking.

The parallelism preference is proposed by Kameyama (1998) to balance the effect of C_b continuity maintained by Centering. It can be stated as two adjacent utterances in discourse seek maximal parallelism. Kameyama argues that parallelism effects on local discourse coherence have been neglected in Centering because Centering focuses on orthogonal "monadic" effects. The parallelism effects on salience weighing and antecedent determining has been noticed by many other researchers (Lappin and Leass 1994; Jiang 2004) and even put into practical pronoun resolution by Kennedy and Boguraev (1996). Therefore, in current research, parallelism preference, specifically represented as the parallelism between the grammatical roles played by the anaphor and the antecedent candidate, is considered as a salience-affecting factor in the C_fs ranking.

- Main/subordinate Consideration

The underlying hypothesis of main/subordinate consideration is that discourse is organized hierarchically. Fox (1987) maintains that "any treatment of anaphora must seek its understanding in the hierarchical structure of the text-type being used as a source of data. Texts may be produced and heard/read in a linear fashion, but they are designed and understood hierarchically".

Miltsakaki (2003) argues that entities evoked in subordinate clauses[3] are less salient than entities evoked in the matrix clauses. Therefore ambiguous pronouns in main clauses tend to be interpreted as referring to the most salient NP in the previous main clause, while ambiguous pronouns in subordinate clauses tend to be interpreted locally. This view also coincides with the claims of Vein Theory advocated by Cristea et al. (1998) and Ide et al. (2000), which proposes that only a subset of the clauses preceding the anaphor are actually relevant to the successful interpretation of the anaphor. This subset (domain of referential accessibility, DRA) is determined by the interaction of the tree hierarchy and whether a clause is a nucleus or a satellite.

2.3 R1-pronouns

R1-pronouns refer to the pronouns covered by Rule 1 of the Centering, i.e., if any C_f is pronominalized, then C_b is. In English Centering literature, although no theory has explicitly specified which types of pronouns are covered by this rule, R1 pronouns are unanimously regarded as third person pronouns. However, in Chinese Centering-based anaphora resolutions,

[1] A syntactic construction where the object occurs before the predicate VP but after the subject

[2] These abbreviated forms are used to label grammatical roles in annotating the texts. See 3.1 for specific denotation.

[3] Subordinate clauses are defined here as tensed adjunct subordinate clauses, specifically, adverbial and relative clauses (Miltsakaki 2003:7).

the preferred realizations of C_bs are zero pronouns (Yeh and Chen 2003; Wang 2004). In order to justify the proper selection of R1-pronouns in current research, zero anaphors and third person pronouns are both taken as R1 pronouns first. The resolution results will offer the ground, at least a statistic ground, to decide which one is more suitable to be R1-pronouns in Centering-based Chinese anaphora resolution.

3. Corpus Annotation and Database Generation

3.1 Corpus annotation

The annotation of our corpus, including the identification criteria of varied grammatical roles, is based on the parse standard of Penn Chinese Treebank (Xue et al. 2005). However in light of the specific purposes of this corpus and economy of computation, we omit the annotation of some syntactic information which is not closely related to our task of anaphora resolution.

Nominal constituents annotated in this research are: zero pronouns (ZP), pronouns (PRON), noun phrases (NP), PROs (PRO), null operators (OP)[4].

The grammatical functions annotated are topic (TPC), subject (SBJ), direct object (OBJ), indirect object (IO), focus (FOC), oblique (OBL), predicate noun phrase (PRDN), head noun in existential construction (EPRDN) and their modifiers.

To examine the effects of structural hierarchy, subordinate clauses are annotated and classified into four types: subject clauses (IP-SBJ), object clauses (IP-OBJ), relative clauses (IP-RET) and adverbial clauses (IP-ADV).

Besides these syntactical properties, the morphological features of NPs, such as number and gender, are also annotated. A set of three possible values is provided for each feature, i.e. single, plural and uncertain for number; male, female and uncertain for gender. Uncertain is used to annotate those fuzzy and hard described features.

At last, five reference types of NPs are recorded: 1) normal: referring to a previous nominal phrase; 2) proposition: referring to a previous proposition; 3) plural: referring to a group of entities; 4) arbitrary: referring to an arbitary member in certain category; 5) cataphora: referring backward. Our research is only devoted to the reference between an anaphor and a previous nominal phrase.

[4] This classification is demanded specifically by our computational inference of the possible antecedent and does not intend to make a scientific claim on the categorization of nominal constituents here.

3.2 Datebase Structure and Generation

We have taken corpus from three discourse genres: first chapter of a modern Chinese novel, eight children's stories, and eight news reports from Penn Chinese Tree Bank. There are totally 31493 words, 5148 NPs, 1476 ZPs and 287 pronouns in our corpus.

A database is designed to accommodate the nominal information of NPs. The parameters set in the database are shown Table 1.

Field name	Data types	Memo
TextForm	Number	1. novel, 2.children stories, 3. News reports
UnitNumber	Number	Serial number of utterances
IPNumber	Number	Serial number of the clause
Position	Number	Serial number of nominals
NPContent	Text	The content of the nominal
NPType	Text	NP, ZP, PRON, PROO…
GFUN	Text	TPC, SBJ, OBJ, OBL…
Number	Number	1 single, 2 plural, 3uncertain
Gender	Number	1 male, 2 female, 3uncertain
RtType	Text	normal, proposition, plural...
CB	BOOL	If the nominal is C_b
SbjIP	BOOL	If the nominal is contained in a subject clause
AdvIP	BOOL	If the nominal is contained in a adverbial clause
ObjIP	BOOL	If the nominal is contained in a object clause
RetIP	BOOL	If the nominal is contained in a relative clause
ResPos	Number	The position of the antecedent computed by the algorithms
AntePos	Number	The position of the antecedent annotated manually

Table 1. Database Structure

4. Resolution Algorithm

4.1 The structure of algorithms

The Centering model is assumed in each algorithm as the fundamental architecture, so is the widely shared anaphora resolution process. Two kinds of anaphora resolution factors are employed: the eliminating factors and "preferential" factors (Mitkov 1999). Three eliminating factors employed here are morphological filter, which deletes from candidate antecedents list those antecedents with disagreeable morphological features; syntactic filter, which involves non-reference between Co-arguments, and a nominal with its modifier unless a marked form is used; and inaccessibility

heritage filter which stipulates that if pronominal A is decided not to be able to refer to B, then it can neither refer to any other referring expressions co-referential with B.

The preference principles in this research are embodies in the examination of different ranking-affecting factors.

4.2 Algorithms designed

We designed six algorithms to examine the effectiveness of different rank-affecting factors and the interactive effect among them. These algorithms are developed according to the natural order from the primitive ranking variations to the more sophisticate ones to examine the effects of gradually complex factors.

Alg.1: Lin. is designed to examine the anaphora resolution when C_f list is ranked according to the occurring order of the discourse entities.

Alg. 2: Grm. is to study the influence of grammatical functions on the ranking of C_fs. It is realized by assigning an arbitrary salience weight to each grammatical role to make the ranking of distinctive grammatical functions possible and also to make the integration with the following rank-affecting factors convenient. Specifically, the salience values assigned to each grammatical function are as follows: TPC: 100; SBJ/ EPRDN: 90; SM/ TM: 80; FOC: 70; OBJ/IO: 60; OM/ IM/ FM/ PM/: 50; OBL/ ADV: 40; OBLM/ ADVM/ APPM: 30. The numerical salience value of each grammatical function is meaningless unless compared with others.

Alg. 3: Para is to test the hypothesis that the antecedents candidates holding the same grammatical function as the anaphor are preferred. The preference is testified by adding extra salience weight, 40, to the discourse entities that play the same grammatical role as that of the anaphor. The weight, 40, is reached empirically by repeated tries on our test corpus for it can just add enough salience weight for objects to weigh heavier than subjects and to be chosen as the preferred antecedent because of the parallel grammatical functions, and not too much to fail other more decisive factors.

Alg. 4: C_b is to testify the preference for the repeatedly referring discourse entities. It is also realized by assigning extra salience weight, 40, to those C_b members in candidate antecedents list. Taking Algorithm 2 as the starting point, Alg. 4 adds the C_b reward after the discourse entities is ranked according to their grammatical roles.

Different from Alg.1 and Alg. 2, which are incompatible in realization, Alg. 5: Para+ C_b is possible to examine the interactive effect of the settings where parallelism consideration is merged with C_b continuity. Alg. 5 is realized by adding C_b reward in Alg. 3.

Alg. 6: Sub is to observe the effect of hierarchical hypothesis on the antecedent determining, which holds that the discourse entities introduced in matrix clauses are more prominent than those introduced in subordinated clauses. This view, when applied to anaphora resolution, is that the anaphor occurring in a matrix clause is preferred to be resolved to a nominal also introduced in a matrix clause, while the anaphor occurring in the subordinated clause is preferred to be resolved locally. We mark all the nominals occurring within a sub-clause as "Sub-clause +" and those occurring in matrix clauses "Sub-clause — ". If the anaphor *P* is found in U_n in a matrix clause, and if all nominals in U_{n-1} are all marked as "Sub-clause +", then the algorithm will jump ahead till the utterance where a nominal marked as Sub-clause " — " is found, and take this utterance as the previous utterance of U_n, then run the algorithm to resolve *P* from in utterance. That is to say, for an anaphor occurring in a main clause, the previous utterance of the anaphor must also be a main clause. When P is within a subordinate clause, we just need to add extra salient weight to those nominals occurring previous in the same subordinate clause.

5. Main Results and Discussion

Table 2 shows a rough tendency that the more factors are taken into consideration; the better results are obtained, although the impacts from different factors are different in degree.

5.1 Effects of Different Ranking-affecting Factors

Figure 1 and 2 show expectedly that Alg.1: Lin is at the bottom in resolution rate, with its least demands of syntactic and discourse information. Considering its comparatively lightness in annotation and computing load, its 84.0% precision rate (1212/1442) in resolving ZP under Udef.1 is rather a satisfying result. It can be accounted for by the fact that Chinese is a SVO language. The surface order of discourse entities, in most cases, coincides with the order of grammatical roles and consequently, is able to reflect the corresponding salience weight.

Therefore, when reliable parsing tools are absent, ranking C_fs in linear order is recommended.

		Novel		Children		News	
		ZP	PRON	ZP	PRON	ZP	PRON
	Total	960	145	415	127	67	6
A1:U1[5]	Pre.[6]	84.4[7]	62.8	86.5	80.3	64.2	50.0
A1:U2	Pre.	68.3	55.2	68.9	66.9	41.8	33.3
A2:U1	Pre.	87.3	64.1	94.9	84.3	88.1	50.0
A2:U2	Pre.	72.1	59.3	80.0	77.2	73.1	66.7
A3:U1	Pre.	88.5	63.4	95.2	82.7	86.6	50.0
A3:U2	Pre.	75.7	64.1	82.4	78.0	70.1	66.7
A4:U1	Pre.	86.1	60.7	93.0	81.9	88.1	50.0
A4:U2	Pre.	70.3	57.2	79.0	78.0	73.1	66.7
A5:U1	Pre.	88.5	63.4	94.7	83.5	86.6	50.0
A5:U2	Pre.	75.8	63.4	81.9	78.8	70.1	66.7
A6:U1	Pre.	93.0	70.3	96.9	83.5	95.5	66.7
A6:U2	Pre.	80.6	64.1	85.1	79.5	88.1	66.7

Table 2 Resolution results

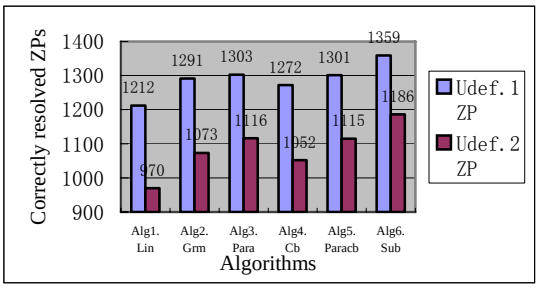


Figure 1. Comparison in ZP resolution

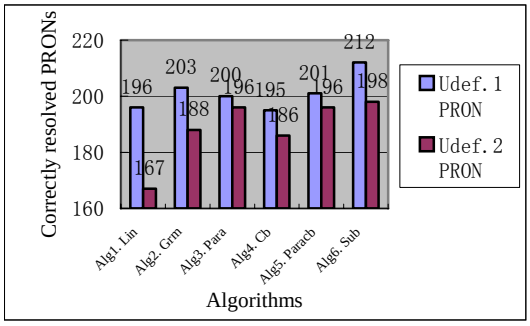


Figure 2. Comparison in PRON resolution

Alg.1 usually fails in the cases where the first mentioned discourse entity is not a subject or a topic, especially the first mentioned discourse entity in the previous utterance. The reason is that intra-utterance referring is usually constrained by syntactic filter, which, to some extend, confines the scope of the possible antecedents, while in inter-utterance resolution, the leftmost constituent is always to be determined as the antecedent no matter what grammatical role it plays. Taking grammatical functions as ranking criterion brings about general improvement in both ZP and PRON resolution. The results prove that grammatical roles are more precise and reliable salience indicators than linear order, and that the salience scale of grammatical roles employed in our research is effective and reasonable. Alg.2 prevails Alg.1 especially in the situations where the first mentioned nominal is in an adjunct position.

The combining in parallelism consideration results in a greater improvement in the corpus segmented by Udef.2: 37 more zero anaphors and 5 pronouns are correctly resolved, and a marginal improvement in the corpus segmented by Udef.1: 12 more zero anaphors are correctly resolved yet at the expense of 3 wrongly resolved pronouns. The facts indicate that parallelism consideration is also a contributive factor in zero anaphor resolution although its effectiveness is marginal compared with that of grammatical roles.

The results indicate that combining in C_b reward is not a desirable idea, for the resolution rate of both ZP and PRON dropped, except for that in news reports. Both Alg.3 and Alg.4 are developed upon the foundation of Alg.2 to examine the parallelism effects and C_b continuity respectively. The results seem to suggest that, from a pure statistic perspective, parallelism of grammatical functions is a more effective factor than C_b continuity in anaphora resolution.

The performance of Alg.5, however, shows that the combination of these two factors does not realize the balanced salience weighing as we expected. The result of Alg.5 is basically the same as that of Alg.3, which proves again that including C_b reward into ranking consideration is not effective. The results, to some extent, justify the doubts of some researchers on the "entity coherence" that entity coherence is not the only means with which the discourse achieves coherence. The tendency to refer to the same discourse entity repeatedly can be overridden by other discourse coherence tendencies such as subjecthood continuity or requirement for maximal parallelism between the grammatical functions performed by the anaphor and its antecedent. The results also show that in the case where C_p (U_{n-1}) is not the C_b(U_{n-1}), the anaphor in U_n chooses C_p(U_{n-1}) as the antecedent in the majority of cases. Shift in discourse attention occurs as naturally as continuity, which is not likely to be something coherent discourse

[5] A1:U1 refers Alg.1 under the segmentation of Udef.1, and the following is on the analogy of this.

[6] Pre refers to precision rate, which is defined as the ratio of the number of correctly resolved anaphors to the number of anaphors which the algorithms attempts to resolve. The correct rates here forth are all precision rate.

[7] Here 84.7 is a percentage number, i.e., 84.7%.

attempts to avoid as much as possible to maintain the coherence of the discourse.

The result of Alg.6 shows that adding hierarchy consideration results in great improvement in ZP and PRON resolution, which proves right the hypothesis that discourse is organized hierarchically and main/subordinate distinction does affect the salience of discourse entities.

	ZP	PRON
	Asymp. Sig[8]	Asymp. Sig.
U1[9]: Alg2 vs. Alg1	0.000	0.143
U1: Alg3 vs. Alg2	0.017	0.375
U1: Alg4 vs. Alg2	0.001	0.008
U1: Alg5 vs Alg3	0.500	1.000
U1: Alg6 vs Alg3	0.000	0.000
U2: Alg2 vs. Alg1	0.000	0.005
U2: Alg3 vs. Alg2	0.000	0.035
U2: Alg4 vs. Alg2	0.004	0.754
U2: Alg5 vs Alg3	1.000	1.000
U2: Alg6 vs Alg3	0.000	0.625

Table 3. Sign test results of algorithms

Table 3 shows the different degree to which these rank-affecting factors influence anaphora resolution.

The results of sign test confirm the findings revealed above that taking grammatical roles as ranking criterion yields positive change to the both ZP and PRON resolution, although the change to PRON resolution under Udef.1 is not significant. Parallelism consideration seems to be more contributive under Udef. 2, which can be accounted for by the fact that parallelism under Udef.2 can help to make up the absence of syntactic filter's effect on the right selection of antecedent. Under Udef.2, the search of antecedent always begins with the NPs at the beginning of the sentence, which, more often than not, do not share the same clause with the anaphor. Therefore syntactic filter in these cases does not work and many objects will be mistakenly resolved to the subjects at the beginning of the sentence. Adding parallelism helps to correct the over-weight salience of the subjects at the sentence initial position under Udef.2 and leads to more correct resolutions of anaphoric objects. The table shows that integration of C_b continuity into ranking is not effective under both Udef.1 and Udef.2. But considering main/subordinate distinction results in significant improvement in resolution of both ZPs and PRONs.

5.2 Effects of utterance parameter

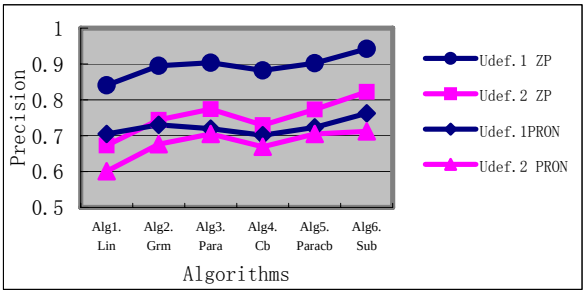


Figure 3. Resolution under Udef.1 and Udef.2

The figure first clearly shows that in both ZP and PRON resolution, the resolution rate under Udef.1 is higher than that under Udef.2. The utterances segmented by Udef.1 can catch shifts of focus by updating the Centering state with comparatively smaller discourse chunk, while utterances defined by Udef.2 are relatively larger and cannot capture the shifts of focus in discourse sensitively, especially unable to reflect the focus change occurring within the utterance. Under Udef.2, the anaphor occurring in the current utterance are always first attempted to be resolved to the discourse entity at the beginning part of the utterance. The conclusion also proves the findings in Poesio et al. (2004) that when utterances are identified with sentences, although it leads to reduction of violations in Constraint 1, it has more violations concerning Rule 1, for the C_b is not updated quickly enough.

Sign test of these two utterance parameters shows that the differences between the two utterance definitions are significant in all resolution of ZP, while insignificant in PRON resolution except in Alg.1 where Udef.1 is significantly better. This fact suggests that pronouns, which do not always refer to the most salient entity in local discourse, are not as sensitive as zero anaphors to the instant updating of Centering state. The might be accounted with the popular hypothesis that more attenuated form the anaphor has, the higher accessibility indictor it is (Arial 1988; Xu 2000). Zero anaphors, with no morphological and phonetic existence, must be interpreted immediately within rather confined discourse scope, while, pronouns, although attenuated compared with full noun phrases, already have morphological features, which help them to establish a comparatively firm co-referential relationships with their

[8] The value of Asymp. Sig is normally used to indicate whether the change in data resulted from alteration of some factor is significant enough from the perspective of possibility. The widely accepted sig. level is 0.05. It is assumed in the statistics field that if the value is larger than 0.05, the change is not significant, vise versa.

[9] U1 refers to Udef.1 and U2 refers to Udef.2.

antecedents, although sometimes the antecedents are spatially far away.

5.3 Data Analysis of R1-pronouns

The result shows clearly that there is a significant and pervasive discrepancy in resolution rate of these two types of pronouns. The precision rate of ZP resolution is much higher than that of PRON resolution. This seems to suggest that in Chinese ZPs are more eligible to assume the role of R1-pronoun than PRONs in Centering-based algorithms.

The reason for low precision rate in pronouns resolution can be attributed first to the fact that pronouns in the discourse can play much more diverse grammatical roles. The referring preference or property of some grammatical roles, such as modifiers, still remains unclear. In our corpus pronouns in modifier positions sometimes refer to the most salient discourse entity but sometimes not, even not to the most salience one in the previous utterance.

Secondly, PRONs in Chinese, compared with ZPs, are more likely to refer to some discourse entity far away. In our corpus, 94.6% of ZPs have their antecedents in current and previous utterances. For PRONs, the percentage is only 74.1%.

Centering model, which aims to compute out the centers of the discourse, i.e., the most salient discourse entities, is more suitable to resolve ZPs, the indicators of the highest accessibility.

6. Theoretical and practical implications

The research results help to reveal some insights into the features of discourse salience and coherence. Our research on the proper setting of utterance shows that smaller discourse chunks are of crucial importance to the accurate resolution of zero pronouns. Zero pronouns in Chinese, compared with pronouns, require quicker updating of Centering or focus state to catch the progress of the discourse. Pronouns, however, are relatively shower in responding to the focus shift in the discourse. Consequently, their resolution is not so crucially affected by the utterance definition.

The study of ranking-affecting factors reveals that grammatical roles played by discourse norminals, compared with their surface order, are more reliable salience indicators. The parallelism tendency observed in some cases is proved to hold although marginally in our research. The negative change in resolution results of Alg.4 and unchanged results of Alg.5 prove that “entity coherence” upheld in Centering does not hold in our corpus, which proves right the argument of Poesio et al. (2004) that “entity coherence” is not the only means with which discourse achieves coherence. Relational and rhetorical coherence are also integral parts in establishment of discourse coherence. The pervasive improvement brought about by the adoption of hierarchical consideration bolsters the view that discourse, although proceeds in linear order, are internally organized in hierarchical structure and there are distinctions of salience between the entities introduced in matrix clauses and those in subordinate clauses. The research on the proper choice of R1-pronouns reinforces the idea that zero pronouns and pronouns in Chinese belong to different types of accessibility markers. Centering model is more suitable for the resolution of zero pronouns, the highest accessibility markers and the norm of anaphora use in Chinese (Xu 2000, 2003; Jiang 2004).

This research also offers specific parameter setting suggestions for the concrete application of Centering Theory to Chinese anaphora resolution. For utterance definition, Udef.1 is strongly recommended for ZP resolution. Ranking in surface order is an approximate alternative of the ranking in grammatical roles when reliable parsing tools are absent. Parallelism consideration is recommended especially when the utterances are identified with sentences. C_b continuity reward is a factor not as effective as expected although it can be a reliable benchmark for measuring discourse coherence. Finally, if possible, the main/subordinate distinction in salience is strongly recommended to be taken into consideration in both ZP and PRON resolution.

Acknowledgements

This research is funded by Youth Scholar Project of Shanghai International Studies University and Outstanding Research Funds of Shanghai.

References

Ariel, M. (1988). Referring and accessibility. *Journal* of Linguistics. 24:65-87

Beaver, D. (2004). The optimization of discourse anaphora. Linguistics and Philosophy. 27(1): 3-56.

Brennan, S., Friedman, M., and Pollard, C. (1987). A Centering approach to pronouns. In Proc. Of the 25th ACL, 155-162, Stanford, USA.

Cristea. D, Ide, N. and Romary, L. (1998). Veins theory: A model of global discourse cohesion and coherence. In Proc. of COLING, 281-285.

Fox. B. (1987). Discourse Structure and Anaphora-Written and Conversational English. Cambridge: Cambridge University Press.

Grosz, B.J., Joshi, A. K., and Weinstein, S. (1995). Centering: A framework for modeling the local coherence of discourse. Computational linguistics, 21(2): 202-225.

Ide, N., and Cristea, D. (2000). A Hierarchical Account of Referential Accessibility. In Proc. of ACL'2000, Hong Kong.

Jiang Ping. (2004). Syntactic and Discourse Features of Zero Anaphora—with Specific Reference to its Resolution in Chinese. PhD thesis, Shanghai International Studies University.

Kameyama, M. (1998). Intra-sentential Centering: A case study. In Walker, M.A., Joshi, A. K., and Prince, E. F., (eds.), *Centering in Discourse*, Ch. 6, 89-112. Ox: Oxford University Press..

Kehler, A. (1997). Current theories of Centering for pronoun interpretation: A critical evaluation. Computational. Linguistics. 23(3):467-475. Marcus, Mitchell P.

Kennedy, C. and Boguraev, B. (1996). Anaphora for Everyone: Pronominal Anaphora Resolution without a Parser. COLING, 113-118.

Lappin, S. Leass, H. (1994). An algorithm for pronominal anaphora resolution. Computational Linguistics, 20(4): 535-561.

Miltsakaki, E. (2003), The Syntax-Discourse Interface Effects of the Main/subordinate Distinction on Attention Structure. PhD thesis. Department of Linguistics, University of Pennsylvania.

Mitkov, R. (1993). Anaphora Resolution and Natural Language Understanding. Unpublished internal report, University Sains Malaysia, Penang.

Nakaiwa, Hiromi, and Satoshi Shirai. 1996. Anaphora resolution of Japanese zero pronouns with deictic reference. In Proceedings of COLING-96, pages 812-817.

Nariyama, Shigeko. 2000. Referent Identification for Ellipted Arguments in Japanese. Ph.D. thesis, The University of Melbourne.

Okumura, Manabu, and Kouji Tamura. 1996. Zero pronoun resolution in Japanese discourse based on centering theory. In Proceedings of COLING-96, pages 871-876.

Poesio, M., Srevenson, R., Eugenio B.D., Hitzeman, J. (2004). Centering: A parametric theory and its instantiations. Computational Linguistics 30: 309–363.

Rambow, O. (1993). Pragmatics aspects of scrambling and topicalization in German. In Proc. of the Workshp on Centering Theory in Naturally-Occurring Discourse, Philadelphia: IRCS.

Strube, M. and Hahn, U. (1999). Functional Centering-grounding referential coherence in information structure. Computational Linguistics, 25(3): 309-344.

Suri, L. Z. and McCoy, K. F., and DeCristofaro, J. (1999). A Methodology for Extending Focusing Frameworks. Computational Linguistics, 25(2): 173-194.

Wang Deliang, (2004). Zero anaphora resolution in Chinese—a Centering-based research. Modern Foreign Language, (4):350-359

Xu, Yulong. (2000) Referring expressions and their accessibility in Chinese and English. Foreign Languages Teaching and Research. (5): 321-328.

Xu, Yulong. (2003). Discourse anaphora resolution: testing a cognitive linguistic model. Journal of Foreign Languages (2): 17-24.

Yeh, C. L. and Chen, Y. J. (2003) Zero anaphora resolution in Chinese with partial parsing based on Centering Theory. In Proc. of NLP-KE03, Beijing, China.

Xue Nianwen, Fei Xia, Fu-Dong Chiou, and Palmer. M. (2005). The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. Natural Language Engineering, 11(2): 198-207.

Author Index

Abekawa, Takeshi, 236
Abney, Steven, 145
Akama, Hiroyuki, 308

Baldwin, Timothy, 40, 129, 333
Bennamoun, Mohammed, 246
Bird, Steven, 22
Buyko, Ekaterina, 163

Cao, Hailong, 113
Cercone, Nick, 191
Cook, Paul, 289
Copestake, Ann, 1
Curran, James, 104, 172, 210, 342

Dale, Robert, 226
Dangsaart, Srisavakon, 191
Dras, Mark, 31, 226
Dridan, Rebecca, 333
Duan, Manjuan, 350

Endo, Tsutomu, 255
Estival, Dominique, 263

Fujii, Seiko, 154
Fukumoto, Fumiyo, 75

Gardiner, Mary, 31
Gaustad, Tanja, 263

Hahn, Udo, 49, 163
Hawker, Tobias, 85
Honnibal, Matthew, 210
Horiguchi, Susumu, 137
Hutchinson, Ben, 263

Iida, Ryu, 120
Ikehara, Satoru, 181
Inui, Kentaro, 120
Isahara, Hitoshi, 113, 281, 298
Ishioroshi, Madoka, 325
Ishizaki, Shun, 94
Iwashita, Kaoru, 255

Jin, Zhihui, 13
Jung, Jaeyoung, 308

Kageura, Kyo, 236
Kanai, Akira, 325
Kanzaki, Kyoko, 298
Kim, Su Nam, 40, 129
Kit, Chunyu, 66
Komachi, Mamoru, 120

Larasati, Septina Dian, 273
Le Nguyen, Minh, 58
Lee, Haejoong, 22
Liu, Wei, 246

Manurung, Ruli, 273
Matsumoto, Yuji, 120
McIntosh, Tara, 342
Miyake, Maki, 308
Mollá, Diego, 317
Mori, Tatsunori, 325
Murakami, Jin'ichi, 181
Murphy, Tara, 172

Naruedomkul, Kanlaya, 191
Nguyen, Cam-Tu, 137
Nguyen, Le-Minh, 137
Nishina, Kikuko, 308

Okamoto, Jun, 94

Paris, Cécile, 226
Patrick, Jon, 218
Pham, Son Bao, 263
Phan, Xuan-Hieu, 137

Radford, Will, 263

Sato, Mitsuru, 325
Schluter, Natalie, 200
Scholz, Bernhard, 172
Shimada, Kazutaka, 255
Shimazu, Akira, 58

Sirinaovakul, Booncharoen, 191
Sornlertlamvanich, Virach, 281
Stevenson, Suzanne, 289
Suzuki, Yoshimi, 75

Tanaka-Ishii, Kumiko, 13
Tokuhisa, Masato, 181
Tomanek, Katrin, 49, 163
Tomuro, Noriko, 298
Tongchim, Shisanu, 281

Uchida, Satoru, 154

Vadas, David, 104
van Genabith, Josef, 200
Van Nguyen, Vinh, 58
van Zaanen, Menno, 317

Wan, Stephen, 226
Wang, Yefeng, 218
Wermter, Joachim, 49
Wong, Wilson, 246

Yang, Li, 145
Yencken, Lars, 13
Yosihasi, Kenji, 308

Zhang, Yitao, 218
Zhang, Yujie, 113
Zhao, Hai, 66